Michael Roduin

3/11/22

**Data Extraction Documentation**

Description:

The beta version for the backend data extraction is updated to include a rudimentary system of data extraction automation locked into only processing .xlsx formal excel files given a set of manually edited parameters. This version requires the manual adding, conversion, and header removal of all excel files before program start to work but presents us with a raw dataset corresponding to the data array included within. The automated data extraction program currently takes a hardcoded list of hospital names with a corresponding .xlsx file under the same name and grabs the entirety of the raw data in its format to generate json objects off of that format. Included with this version is the beginning of a number of data visualization tools to be used to explore the data in order to provide a patch in the future corresponding to identify and convert the output of the documents to a single format to be used as a final database system.

Features:

* Automated processing of documents that follow a standard format and location in the file system.
* Raw Json file output corresponding directly to what is in the documents scanned
* Rudimentary Data manipulation system to modify thew json objects before output
* Initial version of data visualization system for data field names
* System to automate data correction systems in future
* System to automate data visualization systems in future
* Initial data package to present to client of what we have found in the documents.

Testing:

All testing for the data extraction section of the project corresponds to manual identification of whether the procedural output generation is working correctly. Due to the fact that there is a standard required format on the hospital’s end in the generation of these documents and that the documents are not in a standard format we have used manual identification of errors and testing on smaller subsets of the data set for testing the working of these documents it is currently deemed unfeasible to develop an automated set of testing software fort the current state of the project and types of systems we are currently interacting with. Data is only grabbed at the current phase of the project and all most data extracted follows a unique format from hospital to hospital so possible testing in the future after exploration phase will be implemented including for errors such as corrupted excel documents and generation of empty json documents due to any errors in the formatting of the hospital documents.

Software Architecture:

A visual representation of the intended full implementation architecture can be seen below. I decided to develop a control system based on isolation of packages related to what process they are expected to achieve. I have one overall controller to handle the calling of other systems to develop a modular implementation method for adding and removing parts in the future as needed and to enable isolated testing on each module should issues arise in testing. The current plan for this is to separate the modules into a Excel Document Extraction package and a Data Modification package. Also, though it is not contained in this document as it does not correspond to any of the data extraction process for the project there will also be an independent data visualization package in the future to handle assisting in future development through enabling of understanding data that was extracted. Back to the Data extraction system diagramed though, the Data Generation Controller Class will handle the timing and calling of the entire stack of required processes overall firstly calling the Extraction controller class tasked with detecting files to extract, doing any excel file modification required, and then handling all of the data extraction systems required to generate json files from the excel file data. These tasks were isolated into a single object to create an independent system to handle all tasks related to the excel files. Following the completion of this system and generation of the json file output, the program returns back to the Data Generation Controller which in tern calls the Data Modification Controller. This then handles all data quality improvement and erroneous data detection systems in the future to finally generate a json file set that will be directly imported into the MangoDB database to then be connected to the frontend.

Diagram

Description automatically generated