Requirements for Sycamore Consistency Checker

School Year 2021/2022

# Motivation

The automation needs for GGSB drives a need for consistent data in the Sycamore application to avoid implementing transformation in various extracted data for the DSD integration, AATG, or registration processes.

Sycamore does not enforce even the most basic formatting rules for some data and therefore we need to identify some of the defects and in some cases and potentially correct them.

# Requirements

|  |  |  |
| --- | --- | --- |
| **#** | **Description** | **Auto** |
| 1 | First Name, Last Name, and Middle Name for students, staff, and parents must not contain leading or trailing spaces | Y |
| 2 | A student’s nick name must not be included in his or her first name.  **Note**: I found an example <First Name> (<Nickname>). | N |
| 3 | Only one space may be used to separate multiple names as part of student, parent, and staff first, last, or middle names. | (Y) |
| 4 | Double names for last names of students, parents, and staff must be separated by a dash ‘-‘ unless the belong to know exceptions.  Such exceptions are ‘*von* <name>’, ‘*van* <name>’, ‘<name 1> *zu* <name 2>’. | N |
| 5 | All staff must define an email with an ending @gssb.org | N |
| 6 | All active students my must be assigned to a class.  **Note**: students that are not assigned to a class are likely no longer enrolled, e.g. they were withdrawn during the school year. | N |
| 7 | All active families must have at least one active child. Active all children that are not inactive **and** are assigned to a class.  **Note**: the family is considered inactive for the year If all children of s family are withdrawn the family. | N |
| 8 | The student’s ‘homeroom teacher’ must be identical to the student’s class teacher.  Assign the student’s class teacher as homeroom teacher if not set correctly or not set at all. | Y |

# Use of Checker Tool

We are in the process refactoring the Python-based tool to extract data from Sycamore using REST services to address also the SDS requirements.

We will reduce validations and auto-corrections in this tool to a minimum because we assume that the validation tool is run first and that corrections are made in Sycamore before we run the data extraction tool.

The checker tool will offer an option to address some of the issues automatically (if possible). This is only supported for low-risk areas.

The checker tool is run and produces a list of errors with detailed descriptions of

* which rule is violated,
* the context of the error (who is the family, parent, student, employee) so that errors can be easily corrected (if applicable), and
* which auto-corrections were performed.

Ideally, the output of the checker tool is in a Spreadhead / CSV file, which sorts the errors by entity (e.g. student, parent, teacher etc).

The command line tool will return with a short summary that includes an error count if some were found.

# Implementation Strategy

We like to continue using Python 3.x for implementing the new Sycamore Checker tool.

For the data extraction tool (Oskar is now working on it) we concluded that we can make use of the PANDAS library, which is a commonly used representation of data sets.

A simple implementation strategy is to use REST API to retrieve all instances of a relevant entity and place it in a PANDAS dataset. We keep the dataset until the end of the program terminates.

PANDAS may also be used to keep track of the errors that were found.

One benefit of PANDAS is that we can perform dataset operations with multiple datasets with one or two statements. For example, we can filter and join datasets with inner or outer joins. This are powerful operations that are declarative in nature and avoid writing our own equivalent logic.

Two rules above determine if a family and students accidentally active. We can join family, student, and class datasets to determine that students are not assigned to a class and no active student exists in a family based on this single derived dataset.

I suspect that using this logic we can minimize required code and keep it fairly modular.

One the issues are recorded in a PANDAS dataset, we can determine if some of them may be auto-corrected.

This requires that the specific record is retrieved and updated through REST calls. Initially, we will only support the homeroom teacher assignment for automatic correction and expand for there. It is important that we do a proper validation of the code with unit tests and code reviews to avoid undesired side effects.