Metadata Checker

Program Documentation

1. Requirements / Problem Description

Research data management services at universities often rely on researchers to provide sufficient metadata when depositing research data in institutional repositories. In practice, metadata quality can vary considerably, especially when data are uploaded by users without technical support. In the context of the Research Data Management (RDM) team of the Vienna University Library, this issue is frequently encountered in the institutional repository PHAIDRA, where incomplete or inconsistent metadata can limit the findability and reusability of research data.

The goal of this program is to support basic metadata quality control for research data deposits by automatically checking a metadata table for common errors. The program is designed as a small, local command-line application and focuses on typical metadata problems observed in everyday RDM support work.

The program takes a CSV file (metadata.csv) as input, in which each row represents one data file and its associated metadata. The program checks whether required metadata fields are present, whether date values follow the format YYYY-MM-DD, whether license values belong to a predefined list of allowed licenses, and whether filenames follow a simple naming convention.

As output, the program generates a new CSV file (report.csv) that contains the original metadata together with an additional column listing any detected issues for each row. In addition, the program prints a short summary to the console indicating how many metadata records contain errors.

2. Design

The program is designed as a small, sequential Python script that processes metadata records stored in a CSV file. Its overall structure follows a simple control flow consisting of reading input data, validating metadata, and writing a report. The main program flow is illustrated in Figure 1.

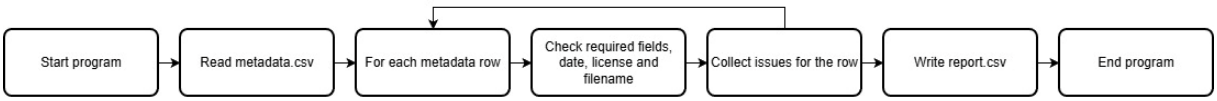

Figure 1: Main program flow of the metadata checker

The program is structured around a central main function, which serves as the entry point and coordinates all processing steps. First, the input file metadata.csv is read using the Python standard library. The file content is stored as a list of rows, where each row is represented as a dictionary mapping column names to values.

The validation logic is implemented using several small helper functions, each responsible for a specific type of check. Separate functions are used to verify the presence of required metadata fields, the format of date values, the validity of license information, and the structure of filenames. Each validation function takes one metadata row as input and returns a list of issues detected for that row. This modular design keeps the code easy to understand and allows individual checks to be modified independently.

The main function iterates over all metadata rows using a loop. For each row, the validation functions are called and their results are combined into a single list of issues. These issue lists are collected for all rows and passed to a report-writing function after the validation loop has finished.

The output is generated by a dedicated function that writes a new CSV file (report.csv). This file contains all original metadata fields as well as an additional column that summarizes the detected issues for each record. Finally, the program prints a short summary to the console indicating how many metadata records contain errors before terminating.

3. Verification / Tests

The program was verified using a small test CSV file containing multiple metadata records with both valid and invalid values. The test data were designed to reflect typical metadata problems encountered in practice, such as missing metadata fields, incorrect date formats, invalid license values, and filenames that do not follow the expected naming convention.

Several test cases were executed by running the program in a local development environment (Visual Studio Code). Records with complete and correctly formatted metadata were used to confirm that no issues were reported for valid input. Additional test records contained empty metadata fields, incorrectly formatted dates, unknown license values, and malformed filenames to ensure that the corresponding validation checks were triggered.

The results of the tests were evaluated by inspecting both the console output and the generated report file (report.csv). For each invalid input record, the expected issues were correctly listed in the output file. Valid records were included in the report without any reported issues. The final console summary correctly indicated the number of records containing errors.

These tests confirm that the program reliably detects common metadata quality issues and produces a consistent and reproducible validation report for the given input data.

AI Statement

An AI-based language model (ChatGPT) was used as a learning and support tool during the completion of this assignment. The AI was used to explain basic Python concepts, to clarify error messages, and to support troubleshooting during program development. In addition, it was used to help structure and phrase documentation sections based on the author's own work and professional context. All code was written, tested, and adapted by the author, and the final program and documentation reflect the author's understanding of the task and its requirements.