Use cases - HFLAV

Table of contents

Definitions

- **Valid data**: Data that meets the requested data type. For example, if valid text is requested, a textual data type must be provided, not a numerical one.
- **Not available data**: This occurs when the server (or the alternative data source being used) is inaccessible to the system, resulting in no data being retrieved.

Actors

ACT-01: Scientist. He uses the available data in the HFLAV Zenodo community to do his work.

Functional requirements

FR-01: The library must allow downloading specific published versions of HFLAV data from Zenodo, identified via DOI.
FR-02: The library should support searching and filtering data.
FR-03: The solution must be implemented as a Python library.
FR-04: The library should be extensible in the future with a CLI.

Use cases

Name	**UC-01: Search relevant records by title inside HFLAV's Zenodo community**
Actors	**ACT-01**
Description	Search all the records that have a specific text inside the Zenodo community
Preconditions	- Have a valid search text
Basic flow	1. The user searches for records giving a specific text 2. The system returns all the records whose title contents that text
Alternative flow	N/A
Exception flow	2. The system returns an exception because the data is not available
Postconditions	- If a cache is available then it will be updated with this new information

Name	**UC-02: Load data from HFLAV's Zenodo community by record id and filename**
Actors	**ACT-01**
Description	Build an object that represents the file data in Python to use for processing
Preconditions	- Have a valid record id - Have a valid filename
Basic flow	1. The user requests the data giving a valid record id and filename 2. The system returns an object that represents that data
Alternative flow	N/A
Exception flow	2. The system returns an exception because the data is not available, the record id or filename are not valid or the data doesn't fit the data template specific version
Postconditions	N/A

Name	**UC-03: Plot data**
Actors	**ACT-01**
Description	Draw a plot that represents the data for analysis and saves the image in the filesystem
Preconditions	- Have a valid plot type - Have a valid plot options
Basic flow	1. The user request the plotting the data giving a valid plot type and options 2. The system returns a visual image which represents the plot 3. The system downloads the image to a specific directory
Alternative flow	N/A
Exception flow	2. The system returns an exception because the plot type or options are not valid 3. The system returns an exception if it doesn't have the right permissions 3. The system returns an exception if it exists a file with the same name
Postconditions	- A file with the plot is downloaded in the path given.

Non functional requirements

1. **Performance:**
    a. **NFR-01 Response Time:** Zenodo queries must complete in under 5 seconds for small datasets (<10MB) and under 30 seconds for large datasets.
    b. **NFR-02 Data Processing:** Data transformation and preparation must efficiently handle datasets up to 1GB in memory. “Efficiently” means low resource consumption and execution under 10 seconds.
    c. **NFR-03 Plot Generation:** Visualizations must render in under 3 seconds for datasets with up to 10,000 data points.
2. **Scalability:**
    a. **NFR-04 Data Handling:** Capacity to process datasets growing 20% annually without performance degradation
3. **Availability:**
    a. **NFR-05 Fault Tolerance:** Library must handle graceful degradation when Zenodo is unavailable
    b. **NFR-06 Automatic Retries**: Retry mechanism with exponential backoff for failed external API connections
    c. **NFR-07 Local Cache**: Cache implementation for frequently accessed data with configurable validity
4. **Maintainability:**
    a. **NFR-08 Automated Testing**: 100% test coverage including unit tests, integration tests, and Zenodo mocks
5. **Usability:**
    a. **NFR-09 Error Messages**: Descriptive error messages oriented toward problem resolution
    b. **NFR-10 Complete Documentation**: Tutorials, usage examples, and API reference available online
6. **Portability**
    a. **NFR-11 Simplified Installation**: Available via PyPI with automatically managed dependencies
    b. **NFR-12 Virtual Environments**: Full compatibility with venv, conda, and pipenv