Google Summer of Code 2026 Final Report

Contributor: Jayanth Vennamreddy
Organization: Apache Software Foundation
Project: CyberShuttle / CS-FileSystem
Repository: https://github.com/cyber-shuttle/CS-FileSystem

Project Goals

For this project, I worked on CS-FileSystem, a user-space filesystem that makes CyberShuttle AI-for-science dataset metadata easier to browse and expose through standard filesystem interfaces. The main goal was to let users inspect dataset metadata through ordinary directories, FUSE mounts, and NFS exports, while also keeping the connector layer flexible enough to support more CyberShuttle datasets later.

Work Completed

- Implemented ATLAS metadata loading from TSV files.
- Represented ATLAS protein entries as directories with a `metadata.json` file for each entry.
- Added support for additional table-backed datasets, including mdCATH, MemProtMD, and GPCRmd.
- Generalized the metadata flow so TSV/CSV-backed datasets can share the same filesystem path.
- Added `index.json` to describe the datasets loaded for a particular run.
- Added an official dataset registry, exposed through `registry.json` and the `datasets/` directory.
- Implemented materialized filesystem export to normal directories.
- Added FUSE mounting support.
- Added NFS server mode for exposing the same dataset tree over NFS.
- Added small sample metadata tables under `examples/` so the project can be tested and demoed without downloading full public datasets.
- Added release packaging support for macOS and Linux binaries.
- Wrote user, developer, and distribution documentation.
- Updated the README with setup instructions, example commands, and links to the new documentation.

Current State

CS-FileSystem can now expose demo and local metadata datasets in three ways:

1. Materialized directory export
2. FUSE mount
3. NFS server

The repository includes small sample tables for reproducible demos. Full public

metadata tables can still be used locally, but they are not committed to the repository because of size and access constraints.

The current implementation focuses on metadata browsing. Large public datasets, restricted scientific datasets, and datasets that require accounts or terms acceptance are represented in the registry, but they are not downloaded automatically during normal demos.

Key Code / Pull Requests

- Main repository: https://github.com/cyber-shuttle/CS-FileSystem
- Main GSoC pull request: https://github.com/cyber-shuttle/CS-FileSystem/pull/2
- Follow-up maintenance work includes the final report, release packaging, and expanded documentation.

Relevant commits:

- https://github.com/cyber-shuttle/CS-FileSystem/commit/c56008d - Add ATLAS metadata filesystem source
- https://github.com/cyber-shuttle/CS-FileSystem/commit/2f038fa - Add materialized ATLAS export mode
- https://github.com/cyber-shuttle/CS-FileSystem/commit/a8b71a9 - Add NFS server mode for ATLAS metadata
- https://github.com/cyber-shuttle/CS-FileSystem/commit/26b03a0 - Add table-backed datasets and filesystem index
- https://github.com/cyber-shuttle/CS-FileSystem/commit/5495651 - Add official dataset registry metadata connector

How To Run

Run the sample materialized export:

```
cargo run --release -- materialize \
  examples/atlas_sample.tsv \
  mdcath=examples/mdcath_sample.tsv \
  memprotmd=examples/memprotmd_sample.tsv \
  gpcrmd=examples/gpcrmd_sample.tsv \
  /tmp/cs_sample_export
```

Inspect the generated filesystem tree:

```
ls /tmp/cs_sample_export
cat /tmp/cs_sample_export/index.json
cat /tmp/cs_sample_export/registry.json
cat /tmp/cs_sample_export/atlas/1r6w_A/metadata.json
```

The same logical tree can also be exposed through FUSE or NFS. The README and `docs/user-guide.md` include the full commands.

Challenges / Lessons Learned

One of the main design challenges was separating dataset-specific parsing from the filesystem tree itself. ATLAS, mdCATH, MemProtMD, and GPCRmd all expose metadata in slightly different shapes, so the implementation needed a common path for table-backed metadata without making every dataset connector depend on ATLAS-specific assumptions.

FUSE and NFS support also made the project more interesting than a simple metadata parser. The filesystem needed stable paths, predictable directory entries, and consistent file contents across materialized exports, mounts, and network serving.

Another important lesson was that scientific datasets often have different access rules, sizes, and licensing requirements. Because of that, the registry needed to describe datasets even when the repository could only include small sample metadata tables for demos.

Remaining Work

- Prepare a Nexus integration demo that shows the filesystem being used inside or alongside Nexus workflows.
- Expand the filesystem from metadata browsing toward deeper CyberShuttle workflow integration.
- Add more dataset-specific connectors beyond table-backed metadata.
- Use the stable filesystem artifact and Nexus demo as the basis for a future paper or technical writeup.
- Continue hardening release artifacts after review by tagging a stable `gsoc-2026-final` snapshot.