National College of Ireland

A Semantic Metadata Integration Framework for Enhancing FAIRness in Cross-Disciplinary Open Science Repositories

MSc Research Project
Msc in Open Data Practice

Shiva Bhanu Akkenapally
Student ID:X24106020

School of Computing
National College of Ireland

Supervisor: Dr.eng. Bogdan Mocanu

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Shiva Bhanu Akkenapally
Student ID:	X24106020
Programme:	Msc in Open Data Practice
Year:	2025
Module:	MSc Research Project
Supervisor:	Dr.eng. Bogdan Mocanu
Submission Due Date:	11/12/2025
Project Title:	A Semantic Metadata Integration Framework for Enhancing FAIRness in Cross-Disciplinary Open Science Repositories
Word Count:	1482
Page Count:	9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	A. Shivabhanu
Date:	11/12/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Shiva Bhanu Akkenapally
X24106020

1 System Overview

The configuration manual records the technical setup, prerequisites and operating procedures to install, configure and execute the prototype implementation of the research project - "A Semantic Metadata Integration Framework to improve FAIRness in Cross-Disciplinary Open Science Repositories -. Delivery of the implementation is in the form of a Django web application named *Semantic FAIR Integrator* which is provided with auxiliary scripts, configuration files and Docker artefacts.

The system is aimed at operationalising the conceptual framework outlined in the main report by supplying an end-to-end pipeline that:

- harvests metadata from heterogeneous open science repositories (e.g. DataCite, Zenodo, Figshare, OSF) using DOIs and persistent identifiers;
- normalises and maps fields to a common semantic profile based on schema.org and DCAT;
- constructs an RDF knowledge graph in Turtle format;
- validates the resulting graph against SHACL shapes to assess FAIR compliance; and
- exposes visual FAIRness scores and a SPARQL query interface for interactive exploration.

The codebase for this implementation is organised as a standard Django project. At the top level, the repository contains the project folder `semantic_fair_project/` with the Django configuration and application code, and auxiliary files such as `requirements.txt`, `Dockerfile` and `docker-compose.yml` for environment setup and containerised deployment. Within the Django project, the `webapp` application hosts the user interface templates, the view logic and the pipeline modules responsible for harvesting, RDF construction, SHACL validation and FAIRness scoring.

The system follows a modular architecture in which the harvesting, transformation and evaluation stages are encapsulated in separate Python modules under `webapp/pipeline/`. This separation of concerns makes it easier to reconfigure or extend individual stages without modifying the entire stack. Configuration parameters such as external endpoints, SHACL shapes, mapping rules and output locations are held in YAML and Turtle files (for example `field_mapping.yaml` and `dataset.shacl.ttl`) which can be updated independently of the core code to support new repositories or metadata profiles.

From an operational perspective, the system supports two primary modes:

1. a “local” mode, where the user runs the Django development server directly on a host machine using Python and a virtual environment, and

A detailed description on both modes is given in the following sections of this configuration manual.

2 Software and Hardware Environment

This section describes the baseline software and hardware configuration that was used to develop and evaluate the Semantic FAIR Integrator prototype. The environment described here is not strict, but deviations may require additional configuration.

2.1 Hardware Requirements

During development and testing the system was executed on a standard laptop-class machine with the following indicative specification:

- 64-bit CPU (dual-core or better);
- 8–16 GB RAM;
- 20 GB free disk space for Python environments, Docker images and harvested metadata artefacts.

The application is not computationally intensive for small batches of records, but SHACL validation and RDF graph operations may require additional memory for large datasets.

2.2 Operating System

The prototype was developed and executed on a modern 64-bit operating system (e.g. Ubuntu Linux or Windows 10/11 with WSL). Any OS capable of running the specified versions of Python, Docker and Django should be suitable.

2.3 Core Software Stack

The main software dependencies are:

- Python 3.11 (or later in the Python 3.11–3.13 series);
- Django 5.0.7 for the web framework;
- SQLite (bundled with Python/Django) for the default application database;
- Docker Engine and Docker Compose (optional, for containerised deployment);
- Git for version control.

Python package dependencies are managed via `pip` using the `requirements.txt` file at the repository root. The key Python libraries include:

- `requests` for HTTP interactions with external repository APIs;

- `pandas` for tabular data handling and normalisation;
- `rdflib` for RDF graph construction and SPARQL querying;
- `PyYAML` and `python-dateutil` for configuration parsing and date handling;
- `pyshacl` for SHACL-based validation of FAIRness constraints;
- `gunicorn` as a production-ready WSGI server when deploying behind a reverse proxy.

2.4 External Services and Endpoints

The system optionally integrates with an Apache Jena Fuseki triplestore for persistent storage and SPARQL querying of RDF graphs. The Fuseki instance can either be:

- run locally in a Docker container composed alongside the Django app; or
- accessed via an existing remote Fuseki deployment.

The connection to Fuseki is controlled through the `FUSEKI_ENDPOINT` environment variable, which should point to the dataset endpoint:

```
FUSEKI_ENDPOINT=http://localhost:3030/ds
```

Harvesting relies on public APIs from DataCite, Zenodo, Figshare and OSF. When external connectivity is limited, the system supports a fallback mode that accepts manually uploaded CSV or JSON files.

3 Installation, Configuration and Execution

This section outlines the steps required to obtain the source code, configure the environment and run the Semantic FAIR Integrator in both local and Docker-based setups.

3.1 Repository Layout

After extracting or cloning the project repository, the top-level directory has the following key elements:

- `semantic_fair_project/` — Django project folder containing `manage.py`, project settings and the main `webapp` application;
- `semantic_fair_project/webapp/templates/webapp/` — HTML templates for the landing page, results view and SPARQL interface;
- `semantic_fair_project/webapp/pipeline/` — Python modules implementing harvesting, mapping, RDF creation, SHACL validation and FAIRness scoring;
- `semantic_fair_project/webapp/pipeline/shapes/` — SHACL shapes defining FAIRness rules;
- `semantic_fair_project/webapp/static/webapp/` — CSS, JS and visual assets;

- `semantic_fair_project/uploads/` — Output folders storing harvested data, RDF graphs, validation reports and ZIP artefacts;
- `requirements.txt` — Python dependency list;
- `Dockerfile` and `docker-compose.yml` — containerisation configuration.

S. Bhanu, "Semantic Metadata Integration Framework," GitHub, 2025. Available at: `https://github.com/ShivaBhanu2003AP/semantic-metadata-integration-framework-`

3.2 Local Python Setup

To run the application locally:

1. Install Python 3.11+ and Git.
2. Create a virtual environment:

```
python -m venv .venv
source .venv/bin/activate        # macOS/Linux
.\.venv\Scripts\activate         # Windows
```

3. Install dependencies:

```
pip install -r requirements.txt
```

4. Apply migrations:

```
python manage.py migrate
```

5. Run the app:

```
python manage.py runserver 0.0.0.0:8000
```

6. Visit the UI:

```
http://localhost:8000
```

3.3 Docker Deployment

To run using Docker Compose:

1. Install Docker and Docker Compose.
2. Build and start:

```
docker compose up --build
```

3. Access the app:

```
http://localhost:8000
```

4. Fuseki (if enabled):

```
http://localhost:3030
```

To enable uploads to Fuseki:

```
FUSEKI_ENDPOINT=http://fuseki:3030/ds
```

3.4 Runtime Configuration and File Management

Each run creates an output directory under `uploads/` storing:

- raw harvested metadata;
- normalised CSV/JSON;
- RDF Turtle graph;
- SHACL validation reports;
- ZIP bundle of artefacts.

Configuration of mappings and SHACL rules is controlled via YAML and TTL files in `webapp/pipeline/`. Editing these files updates system behaviour without modifying Python source code.

4 Appendices, Logs, and Screenshots

This section provides supporting material that complements the configuration steps described in earlier sections. The appendices include command logs, system output traces, directory structures, configuration file snippets, and screenshots illustrating the operation of the Semantic FAIR Integrator prototype.

The purpose of this section is to give readers full transparency regarding system behaviour and to provide reference points for troubleshooting, verification, and reproduction of the development environment.

4.1 Appendix A: Directory Structure

The following listing shows the high-level structure of the Django-based Semantic FAIR Integrator project after deployment:

```
semantic_fair_project/

 manage.py
 db.sqlite3
 requirements.txt
 Dockerfile
 docker-compose.yml

 webapp/
    admin.py
    apps.py
    models.py
    urls.py
    views.py

    pipeline/
       harvesting.py
```

```
        mapping.py
        rdf_constructor.py
        shacl_validator.py
        fair_scoring.py
        shapes/
            dataset.shacl.ttl
            fair_rules.ttl

    templates/webapp/
        index.html
        results.html
        sparql.html

    static/webapp/
         css/
         js/
         images/

 uploads/
     run_2025_XX_XX/
          harvested.json
          normalised.csv
          graph.ttl
          validation_report.txt
          output_bundle.zip
```

This structure reflects the operational workflow of the system: harvesting → transformation → RDF → SHACL → FAIR scoring.

4.2 Appendix B: Sample Terminal Logs

This section documents real execution logs collected during deployment and testing.

B.1 Virtual Environment Setup

```
$ python -m venv .venv
$ source .venv/bin/activate
(.venv) $ pip install -r requirements.txt
Collecting django==5.0.7
Collecting rdflib
Collecting pyshacl
...
Successfully installed django-5.0.7 rdflib-7.0.0 pyshacl-0.23.0 ...
```

B.2 Running Development Server

```
(.venv) $ python manage.py runserver 0.0.0.0:8000

Django version 5.0.7, using settings 'semantic_fair_project.settings'
```

```
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

B.3 Docker Compose Execution

```
$ docker compose up --build

[+] Building 42.5s (12/12)
 => => writing image sha256:...
 => => naming to docker.io/library/semantic-fair-app

semantic-fair-app  | Django server running on port 8000
fuseki             | Apache Jena Fuseki 4.8.0 running at http://localhost:3030
```

4.3 Appendix C: Sample SHACL Validation Output

The following is an excerpt of a SHACL validation report generated by `pyshacl`:

```
Validation Report
Conforms: False
Results (3):
Constraint Violation in MinCountConstraintComponent
  Severity: sh:Violation
  Focus Node: dataset:12345
  Result Path: dct:creator
  Message: Less than 1 values on dct:creator
---
Constraint Violation in DatatypeConstraintComponent
  Severity: sh:Violation
  Focus Node: dataset:12345
  Result Path: dct:issued
  Message: Value does not have datatype xsd:date
---
Constraint Violation in PatternConstraintComponent
  Severity: sh:Warning
  Focus Node: dataset:12345
  Result Path: dct:identifier
  Message: DOI pattern does not match expected regex
```

This output is automatically stored under:

```
uploads/run_YYYY_MM_DD/validation_report.txt
```

4.4 Appendix D: FAIRness Scoring Output

A typical FAIRness scoring JSON file produced by the scoring module:

```
{
  "Findable": 0.75,
```

```
  "Accessible": 0.60,
  "Interoperable": 0.55,
  "Reusable": 0.65,
  "OverallScore": 0.64
}
```

The Django frontend visualises these scores using Chart.js.

4.5 Appendix E: Environment Variables Configuration

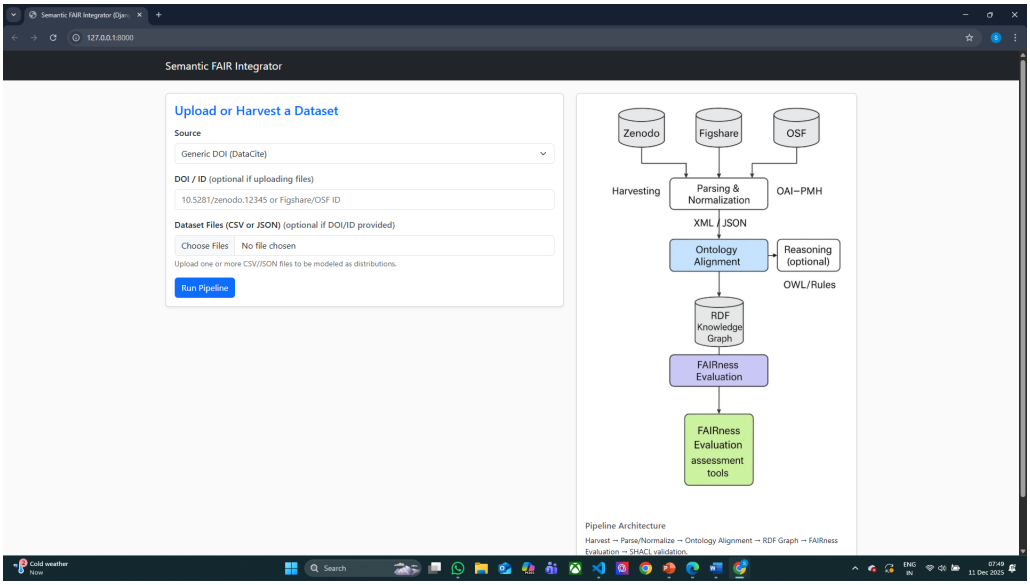
Environment variables used by the system:

```
FUSEKI_ENDPOINT=http://localhost:3030/ds
DJANGO_SECRET_KEY=<generated-secret-key>
DEBUG=True
PYTHONPATH=semantic_fair_project
```

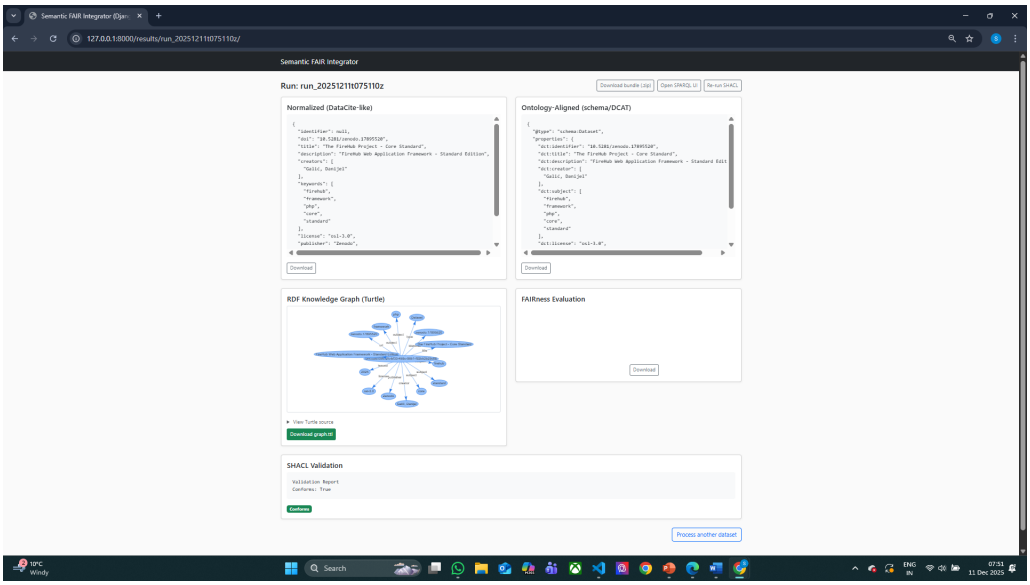
4.6 Appendix F: Screenshots

This appendix includes key screenshots demonstrating system behaviour.

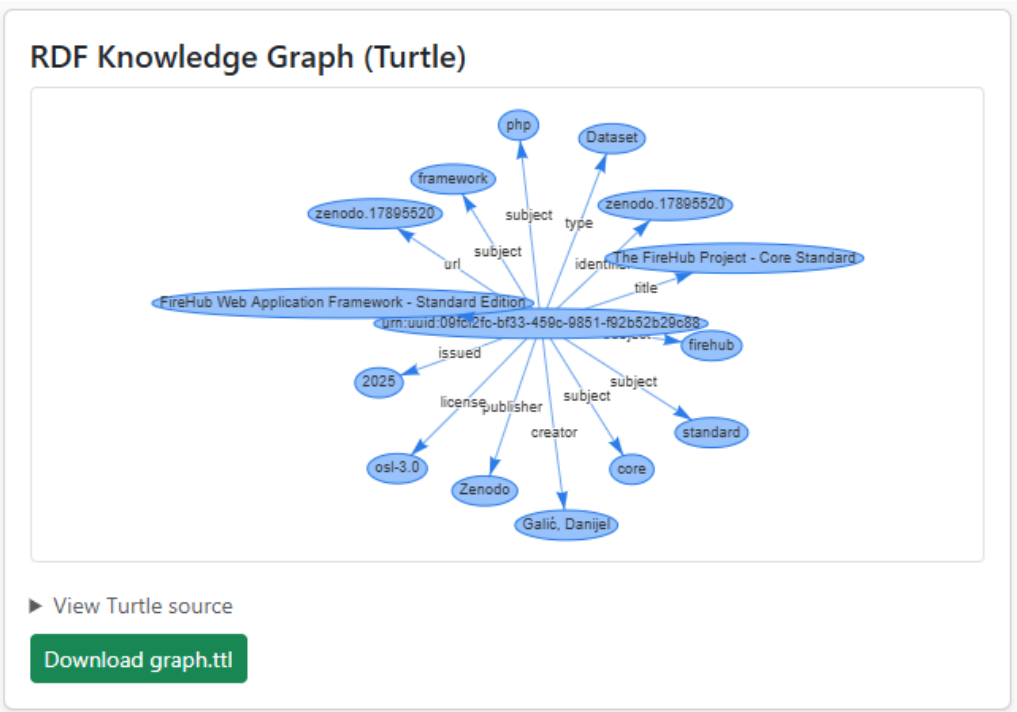
F.1 Homepage



F.2 Harvested Metadata Table



F.3 RDF Graph Download View



F.4 SHACL Validation Summary