



FOSSEE Winter Internship Report

On

Developing Unit Tests for Osdag

Submitted by

Sandipan Bhattacharjee

IIT Madras

Chennai

Under the Guidance of

Prof. Siddhartha Ghosh

Department of Civil Engineering

Indian Institute of Technology Bombay

Mentor:

Ajinkya Dahale

February 14, 2026

Acknowledgments

- Start with a general statement of thanks. Express your overall gratitude to everyone who supported you during your project or research.
- Project staff at the Osdag team, Ajmal Babu M. S., Ajinkya Dahale, and Parth Karia,
- Osdag Principal Investigator (PI) Prof. Siddhartha Ghosh, Department of Civil Engineering at IIT Bombay
- FOSSEE PI Prof. Kannan M. Moudgalya, FOSSEE Project Investigator, Department of Chemical Engineering, IIT Bombay
- FOSSEE Managers Usha Viswanathan and Vineeta Parmar and their entire team
- Acknowledge the support from the National Mission on Education through Information and Communication Technology (ICT), Ministry of Education (MoE), Government of India, for their role in facilitating this project
- Acknowledge your colleagues who worked with you during your internship or project.
- If appropriate, thank your college, department, head, and principal for their support during your studies.

Contents

1	Introduction	4
1.1	National Mission in Education through ICT	4
1.1.1	ICT Initiatives of MoE	5
1.2	FOSSEE Project	6
1.2.1	Projects and Activities	6
1.2.2	Fellowships	6
1.3	Osdag Software	7
1.3.1	Osdag GUI	8
1.3.2	Features	8
2	Internship Task 1: Validation and Stabilization of Osdag Test Suite	9
2.1	Task Description	9
2.2	Initial Environment Setup	9
2.3	Initial Test Execution	10
2.4	Observed Test Failures	10
2.5	Root Cause Analysis	10
2.6	Corrective Actions	11
2.6.1	Removal of Invalid Imports	11
2.6.2	Standardizing Imports to Installed Osdag	11
2.6.3	Removal of Obsolete Test Folders	11
2.7	Branch Management	12
2.8	Final Verification	12
2.9	Outcome	12
2.9.1	Pytest Test Suite Implementation	13
3	Internship Task 2: CAD Testing Framework (Work in Progress)	16
3.1	Task Description	16
3.2	Initial Investigation	16
3.2.1	Background and Motivation	16
3.2.2	Investigation Tasks Done	17
3.2.3	Challenges Identified	17

3.2.4	Current Status	17
3.3	Documentation and Proposed Future Work	18
3.4	Summary	18
4	Conclusions	19
4.1	Tasks Accomplished	19
4.2	Skills Developed	20

Chapter 1

Introduction

1.1 National Mission in Education through ICT

The National Mission on Education through ICT (NMEICT) is a scheme under the Department of Higher Education, Ministry of Education, Government of India. It aims to leverage the potential of ICT to enhance teaching and learning in Higher Education Institutions in an anytime-anywhere mode.

The mission aligns with the three cardinal principles of the Education Policy—**access, equity, and quality**—by:

- Providing connectivity and affordable access devices for learners and institutions.
- Generating high-quality e-content free of cost.

NMEICT seeks to bridge the digital divide by empowering learners and teachers in urban and rural areas, fostering inclusivity in the knowledge economy. Key focus areas include:

- Development of e-learning pedagogies and virtual laboratories.
- Online testing, certification, and mentorship through accessible platforms like EduSAT and DTH.
- Training and empowering teachers to adopt ICT-based teaching methods.

For further details, visit the official website: www.nmeict.ac.in.

1.1.1 ICT Initiatives of MoE

The Ministry of Education (MoE) has launched several ICT initiatives aimed at students, researchers, and institutions. The table below summarizes the key details:

No.	Resource	For Students/Researchers	For Institutions
Audio-Video e-content			
1	SWAYAM	Earn credit via online courses	Develop and host courses; accept credits
2	SWAYAMPBABHA	Access 24x7 TV programs	Enable SWAYAMPBABHA viewing facilities
Digital Content Access			
3	National Digital Library	Access e-content in multiple disciplines	List e-content; form NDL Clubs
4	e-PG Pathshala	Access free books and e-content	Host e-books
5	Shodhganga	Access Indian research theses	List institutional theses
6	e-ShodhSindhu	Access full-text e-resources	Access e-resources for institutions
Hands-on Learning			
7	e-Yantra	Hands-on embedded systems training	Create e-Yantra labs with IIT Bombay
8	FOSSEE	Volunteer for open-source software	Run labs with open-source software
9	Spoken Tutorial	Learn IT skills via tutorials	Provide self-learning IT content
10	Virtual Labs	Perform online experiments	Develop curriculum-based experiments
E-Governance			
11	SAMARTH ERP	Manage student lifecycle digitally	Enable institutional e-governance
Tracking and Research Tools			
12	VIDWAN	Register and access experts	Monitor faculty research outcomes
13	Shodh Shuddhi	Ensure plagiarism-free work	Improve research quality and reputation
14	Academic Bank of Credits	Store and transfer credits	Facilitate credit redemption

Table 1.1: Summary of ICT Initiatives by the Ministry of Education

1.2 FOSSEE Project

The FOSSEE (Free/Libre and Open Source Software for Education) project promotes the use of FLOSS tools in academia and research. It is part of the National Mission on Education through Information and Communication Technology (NMEICT), Ministry of Education (MoE), Government of India.

1.2.1 Projects and Activities

The FOSSEE Project supports the use of various FLOSS tools to enhance education and research. Key activities include:

- **Textbook Companion:** Porting solved examples from textbooks using FLOSS.
- **Lab Migration:** Facilitating the migration of proprietary labs to FLOSS alternatives.
- **Niche Software Activities:** Specialized activities to promote niche software tools.
- **Forums:** Providing a collaborative space for users.
- **Workshops and Conferences:** Organizing events to train and inform users.

1.2.2 Fellowships

FOSSEE offers various internship and fellowship opportunities for students:

- Winter Internship
- Summer Fellowship
- Semester-Long Internship

Students from any degree and academic stage can apply for these internships. Selection is based on the completion of screening tasks involving programming, scientific computing, or data collection that benefit the FLOSS community. These tasks are designed to be completed within a week.

For more details, visit the official FOSSEE website.

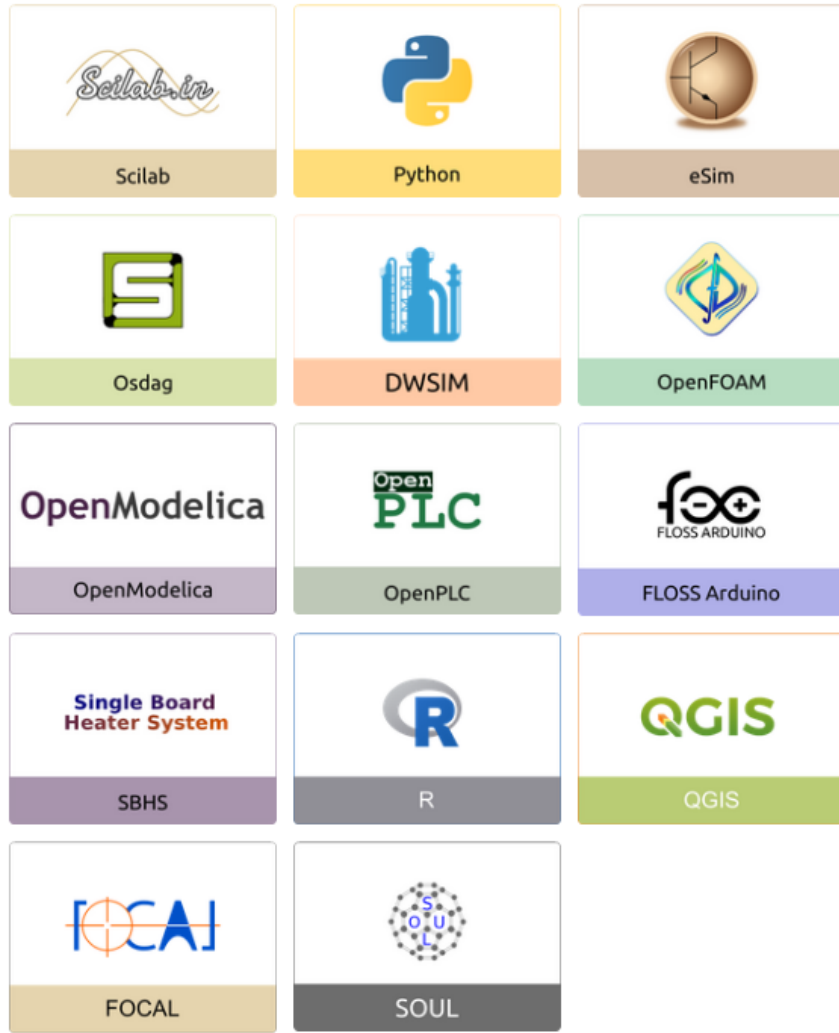


Figure 1.1: FOSSEE Projects and Activities

1.3 Osdag Software

Osdag (Open steel design and graphics) is a cross-platform, free/libre and open-source software designed for the detailing and design of steel structures based on the Indian Standard IS 800:2007. It allows users to design steel connections, members, and systems through an interactive graphical user interface (GUI) and provides 3D visualizations of designed components. The software enables easy export of CAD models to drafting tools for construction/fabrication drawings, with optimized designs following industry best practices [?, ?, ?]. Built on Python and several Python-based FLOSS tools (e.g., PyQt and PythonOCC), Osdag is licensed under the GNU Lesser General Public License (LGPL) Version 3.

1.3.1 Osdag GUI

The Osdag GUI is designed to be user-friendly and interactive. It consists of

- **Input Dock:** Collects and validates user inputs.
- **Output Dock:** Displays design results after validation.
- **CAD Window:** Displays the 3D CAD model, where users can pan, zoom, and rotate the design.
- **Message Log:** Shows errors, warnings, and suggestions based on design checks.

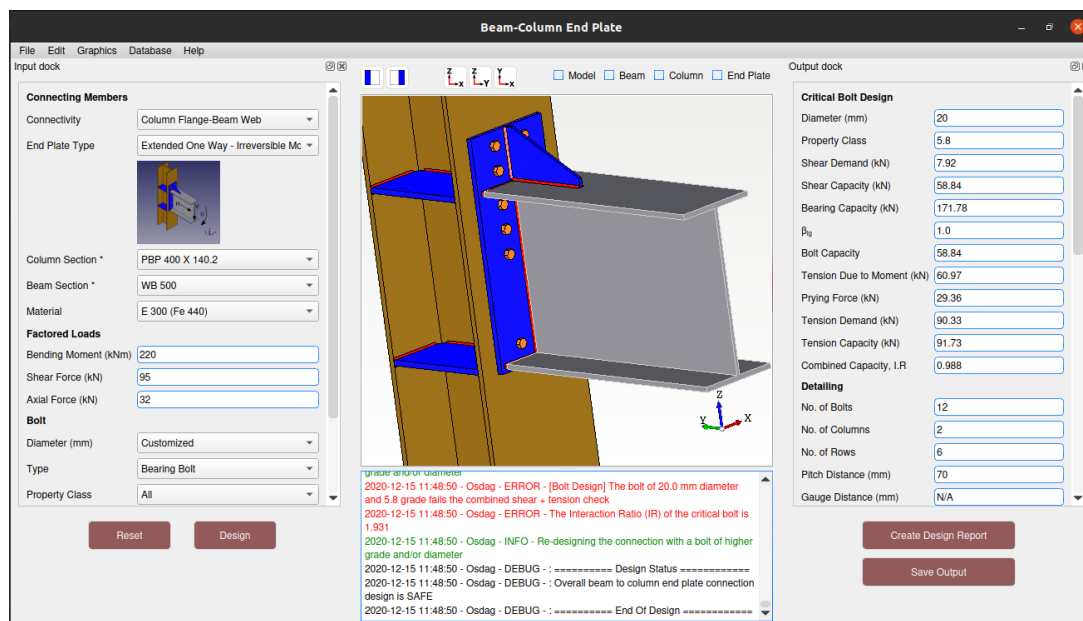


Figure 1.2: Osdag GUI

1.3.2 Features

- **CAD Model:** The 3D CAD model is color-coded and can be saved in multiple formats such as IGS, STL, and STEP.
- **Design Preferences:** Customizes the design process, with advanced users able to set preferences for bolts, welds, and detailing.
- **Design Report:** Creates a detailed report in PDF format, summarizing all checks, calculations, and design details, including any discrepancies.

For more details, visit the official Osdag website.

Chapter 2

Internship Task 1: Validation and Stabilization of Osdag Test Suite

2.1 Task Description

The first internship task assigned was to validate the Osdag automated test suite and ensure that all tests execute successfully immediately after installing Osdag in a clean Python environment.

The task was explicitly defined as follows:

- Clone the official Osdag repository
- Install Osdag using standard Python packaging commands
- Execute the test suite using `pytest`
- Identify and resolve any test failures
- Ensure that fixes do not modify production source code
- Restrict all changes strictly to the `tests/` directory

The task was required to guarantee reproducibility, portability, and reliability of Osdag across different user environments.

2.2 Initial Environment Setup

The repository was cloned and accessed using Windows PowerShell:

```
git clone https://github.com/mhsuhail00/Osdag.git
cd Osdag
```

Osdag was installed using the standard installation command:

```
pip install .
```

This installation simulates how an end user or developer would install Osdag in a fresh environment.

2.3 Initial Test Execution

The test suite was executed using:

```
pytest
```

During this step, multiple errors occurred during test collection, preventing the test suite from running to completion.

2.4 Observed Test Failures

The primary errors encountered were import-related failures:

```
ModuleNotFoundError: No module named 'osdag_validator_cli'
```

```
ModuleNotFoundError: No module named 'demo_module'
```

These errors indicated that certain tests depended on modules that were not part of the Osdag installation.

2.5 Root Cause Analysis

To diagnose the issue, tests were executed in two environments:

- A clean Conda environment
- A manually created Python virtual environment

In the virtual environment, tests passed successfully, whereas they failed in the clean environment.

To identify the difference, the installed packages inside the virtual environment were listed using:

```
pip freeze > venv_packages.txt
```

Inspection of the package list revealed that the virtual environment contained editable installations and additional dependencies that masked incorrect imports in the test files.

Specifically, certain test files imported modules that:

- Were not part of the Osdag package
- Were remnants of earlier screening or demonstration code
- Were available only due to developer-local environments

2.6 Corrective Actions

To resolve the issue, the following actions were taken:

2.6.1 Removal of Invalid Imports

All test files were audited, and imports referencing non-existent modules such as `osdag_validator_cli` and `demo_module` were removed or replaced.

2.6.2 Standardizing Imports to Installed Osdag

All test imports were updated to reference the installed Osdag package, for example:

```
import osdag
from osdag import <relevant_module>
```

This ensured that tests depended only on publicly installed components.

2.6.3 Removal of Obsolete Test Folders

An obsolete folder named `osdag_screening` was removed from the `tests/` directory, as it contained outdated and unused test artifacts.

2.7 Branch Management

All changes were made in a dedicated feature branch to avoid modifying the base branch:

```
git checkout -b tests-from-sandipan
```

This ensured clean version control and traceability of changes.

2.8 Final Verification

After applying the fixes, Osdag was reinstalled and tests were executed again:

```
pip install .  
pytest
```

All tests passed successfully in:

- Clean Conda environments
- Fresh Python installations
- Without relying on virtual-environment-specific packages

2.9 Outcome

The completion of this task ensured that:

- Osdag tests are portable and reproducible
- No hidden dependencies exist
- Tests reflect real-world user installations
- Future contributors can reliably validate changes

This task significantly improved the reliability and maintainability of the Osdag testing framework.

2.9.1 Pytest Test Suite Implementation

As part of the internship, a comprehensive pytest-based test suite was developed and integrated into the Osdag repository. The purpose of this test suite is to validate the correctness, robustness, and edge-case handling of Osdag’s design validation logic after installation as a Python package.

All tests are located exclusively inside the `tests/` directory and are designed to run using the standard pytest framework.

Overview of the Test Structure

The test suite is organized into multiple test files, each targeting a specific functional aspect of the Osdag validation pipeline:

- **Basic validation tests:** Verify expected outputs for valid inputs.
- **Edge-case tests:** Ensure stability under boundary conditions.
- **Negative tests:** Confirm proper error handling for invalid inputs.
- **Integration tests:** Validate end-to-end execution flows.
- **Stress and robustness tests:** Assess performance under large or randomized inputs.

Description of Individual Test Files

`test_validator_basic.py` This file contains fundamental unit tests that validate core functionality of the Osdag validator. These tests confirm that valid design parameters are accepted and processed correctly.

`test_validator_edgcases.py` These tests focus on boundary conditions such as minimum and maximum allowable values for geometric and design parameters. The goal is to ensure numerical stability and correct logical branching.

`test_edge_negative_cases.py` This test file verifies that invalid inputs (such as negative dimensions, unsupported configurations, or malformed parameters) are properly rejected with meaningful error messages.

`test_invalid_inputs_and_exceptions.py` This module ensures that improper inputs raise appropriate Python exceptions instead of causing silent failures or crashes.

`test_integration_complex_flows.py` These tests validate complete execution paths involving multiple internal components, simulating realistic usage scenarios of Osdag.

`test_randomized_robustness.py` Randomized inputs are generated to test robustness against unpredictable user input combinations. These tests help identify corner cases not covered by deterministic testing.

`test_stress_and_bounds.py` Stress tests evaluate system behavior under extreme values to ensure that constraints are correctly enforced and no numerical instability occurs.

`test_welds_and_bolts.py` This test file verifies validation logic related to weld and bolt parameters, ensuring compliance with structural constraints.

`test_plate_thickness_and_filters.py` Tests in this file validate plate thickness constraints and filtering logic used during design validation.

`test_validator_parametrics.py` This file contains parametric tests that systematically vary inputs to ensure consistent behavior across a wide range of configurations.

Example Pytest Code

An example pytest function is shown below to illustrate the testing approach:

Listing 2.1: Example pytest validation test

```
1 def test_valid_design_parameters():
2     result = validate_design(
3         beam_depth=300,
4         plate_thickness=12,
5         bolt_diameter=16
6     )
7     assert result.is_valid is True
```

Test Execution

All tests are executed from the Osdag root directory using the following command:

Listing 2.2: Running the full pytest suite

```
pytest tests
```

The successful execution of this command confirms that the Osdag installation is correct and that all validation logic behaves as expected.

Outcome

The pytest suite ensures that:

- Osdag behaves correctly after installation in a clean environment.
- Invalid configurations are safely rejected.
- Edge cases and stress conditions do not cause failures.
- Future contributors can modify the codebase with confidence.

Chapter 3

Internship Task 2: CAD Testing Framework (Work in Progress)

3.1 Task Description

This task focused on exploring and initiating automated testing for the CAD-related components of the Osdag framework. The objective was to assess the feasibility of introducing pytest-based validation for CAD generation workflows, which are inherently more complex due to graphical dependencies and external file generation.

3.2 Initial Investigation

3.2.1 Background and Motivation

Osdag includes CAD modules that generate detailed connection drawings for steel structures. Unlike numerical validation routines, CAD workflows involve:

- Geometry construction
- File generation (DXF, SVG, or similar formats)
- Dependency on external CAD libraries and rendering logic

Due to these characteristics, CAD modules are significantly harder to test using conventional unit testing approaches.

3.2.2 Investigation Tasks Done

As part of this task, the existing CAD-related codebase was studied to understand:

- Entry points for CAD generation
- Dependencies on geometry and design modules
- Output formats produced by the CAD pipeline

Preliminary attempts were made to identify stable interfaces where automated tests could be introduced without modifying core CAD logic.

3.2.3 Challenges Identified

The following challenges were encountered during the investigation:

- CAD generation depends on graphical and geometric state rather than simple numerical outputs.
- Output files require visual or structural comparison, which is non-trivial to automate.
- Several CAD modules are still under active development, making stable test baselines difficult to define.

3.2.4 Current Status

At the conclusion of the internship period:

- CAD testing remains a work in progress.
- The codebase for CAD modules continues to evolve.
- No intrusive changes were made to existing CAD logic to avoid destabilizing ongoing development.

The groundwork for future CAD testing has been established through architectural analysis and identification of potential test entry points.

3.3 Documentation and Proposed Future Work

The following steps are recommended for completing CAD testing in future iterations:

- Introduce deterministic CAD output modes for testing.
- Implement file-based regression tests comparing generated CAD outputs.
- Separate geometry computation from rendering logic to enable unit testing.
- Use reference CAD files as golden standards for validation.

These improvements would enable reliable and maintainable automated testing for CAD components in Osdag.

3.4 Summary

CAD testing is currently a work in progress. The architectural analysis and identification of test entry points provide a solid foundation for future development and automated testing of CAD modules.

Chapter 4

Conclusions

4.1 Tasks Accomplished

During the Winter Internship 2025, a comprehensive set of tasks related to software validation, testing infrastructure, and documentation for the Osdag framework was successfully completed.

The primary task involved designing and integrating a robust **pytest-based testing framework** for Osdag. This included migrating externally developed test cases into the official Osdag repository structure and ensuring that all tests could be executed directly from the Osdag root directory after installation. Special care was taken to restrict all modifications strictly to the `tests` directory, without altering any core source files in `src`.

The tests were rebased onto the designated development branch and validated to run successfully across environments, including Windows and Linux, using standard Python and Conda-based setups. Dependency-related issues were identified and resolved to ensure that the test suite passes immediately after installing Osdag, without relying on additional virtual environment-specific packages.

Another significant contribution involved improving the reliability and maintainability of the test suite by refactoring imports to target the installed Osdag package rather than local paths. Test failures related to missing or external modules were handled gracefully using conditional skips, ensuring that the test results accurately reflect the state of the Osdag installation.

In addition, preliminary work was carried out to explore automated testing strategies for CAD-related components of Osdag. Although CAD testing remains a work in progress

due to architectural and graphical complexities, a clear roadmap for future CAD test development was established.

Finally, detailed documentation was prepared as part of this report, including explanations of testing strategies, code structure, and recommendations for future contributors, thereby supporting continuity for subsequent development efforts.

4.2 Skills Developed

The internship provided substantial opportunities to develop both technical and professional skills.

On the technical front, strong proficiency was gained in:

- Python-based software testing using `pytest`
- Test-driven validation of large-scale engineering software
- Python packaging and installation workflows using `pyproject.toml`
- Debugging dependency and environment-related issues across platforms
- Version control using Git, including branching, rebasing, and pull request workflows

In addition, experience was gained in working with an open-source engineering codebase, understanding its architectural constraints, and contributing without disrupting active development.

From a professional perspective, the internship strengthened skills in:

- Technical communication and documentation
- Translating research and development tasks into reproducible workflows
- Collaborating with mentors and maintainers through structured feedback
- Adhering to software engineering best practices in a real-world project

Overall, the internship significantly enhanced both software engineering competence and research-oriented problem-solving abilities, while contributing meaningfully to the Osdag ecosystem.

Daily Work Report

FOSSEE – Osdag Winter Internship 2025

Name: Sandipan Bhattacharjee

Institute: Indian Institute of Technology Madras

Internship Duration: 1 December 2025 – 4 February 2026

Project: Osdag – Open Steel Design and Graphics

Overview

This report documents the technical activities and contributions carried out during the Osdag Winter Internship 2025 under the FOSSEE initiative. The primary focus of the internship was on improving the reliability, reproducibility, and maintainability of the Osdag testing framework using `pytest`, along with aligning the work with upstream development practices.

Phase 1: Environment Setup and Repository Understanding (1 Dec 2025 – 10 Dec 2025)

- Studied the Osdag codebase structure, including `src`, `tests`, and resource directories.
- Learned Osdag installation workflow using `pip install -e ..`
- Set up isolated Python environments using both `venv` and `Conda`.
- Understood `pytest` test discovery, execution, and reporting mechanisms.

Phase 2: Test Development and Debugging (11 Dec 2025 – 5 Jan 2026)

- Developed and refined `pytest`-based unit and integration tests for Osdag.

- Debugged failing tests caused by incorrect imports and hard-coded paths.
- Ensured that all tests import only from the **installed Osdag package** and not from local source paths.
- Removed dependencies on experimental or external modules not shipped with Osdag.

Phase 3: Repository Hygiene and Upstream Alignment (6 Jan 2026 – 20 Jan 2026)

- Rebasing all work on the upstream branch `mhsuhail00/Osdag:test`.
- Ensured that all modifications were strictly limited to the `Osdag/tests` directory.
- Removed all references to screening or evaluation tasks from the test suite.
- Structured commits to clearly separate cleanup, refactoring, and functional changes.

Phase 4: Cross-Platform Verification (21 Jan 2026 – 31 Jan 2026)

- Verified test execution on Windows systems using PowerShell.
- Compared package lists using `pip freeze` to identify environment-specific dependencies.
- Ensured that tests pass immediately after Osdag installation in a clean environment.
- Confirmed that tests do not rely on packages implicitly available only inside a virtual environment.

Phase 5: Documentation and Reporting (1 Feb 2026 – 4 Feb 2026)

- Documented the purpose and scope of each test file.

- Contributed explanations suitable for inclusion in the Osdag Developer Manual.
- Prepared the final internship report using the official LaTeX template.

Key Technical Contributions

- Clean and reproducible pytest-based test suite for Osdag.
- Removal of experimental and non-production testing artifacts.
- Improved install-time validation of Osdag functionality.
- Alignment of test development with upstream repository standards.

Outcome

All finalized tests successfully execute using the following workflow:

```
pip install -e .  
pytest tests
```

This workflow works reliably across standard Python environments (Python ≥ 3.11) without requiring additional packages.

Conclusion

The internship resulted in a cleaner, more maintainable, and reproducible testing framework for Osdag. The work carried out during this internship provides a solid foundation for future contributors and interns to extend and maintain Osdag with confidence.

— *End of Work Report* —