FOSSEE Summer Fellowship Report

On

Unit Testing and GUI Development for Osdag

Submitted by

Prince Sahu

4th Year B.Tech Student, Department of Computer Science and Enginnering

Vellore Institute of Technology

Bhopal

Under the Guidance of

Prof. Siddhartha Ghosh

Department of Civil Engineering

Indian Institute of Technology Bombay

Mentors:

Ajmal Babu M S

Parth Karia

Ajinkya Dahale

August 19, 2025

Acknowledgments

- I would like to express my sincere gratitude to all those who supported and guided me throughout my internship with Osdag. This experience has been invaluable for my professional development in structural engineering software.
- My deepest appreciation goes to the entire Osdag team for their mentorship, particularly to Ajmal Babu M. S., Ajinkya Dahale, and Parth Karia for their patient guidance and technical expertise during my project work.
- I am honored to acknowledge the leadership of Prof. Siddhartha Ghosh, Principal Investigator of Osdag from the Department of Civil Engineering at IIT Bombay, for creating this impactful open-source initiative.
- Special thanks to Prof. Kannan M. Moudgalya, FOSSEE Project Investigator from the Department of Chemical Engineering at IIT Bombay, for his vision in supporting open-source engineering education.
- I gratefully acknowledge the support from FOSSEE managers Usha Viswanathan and Vineeta Parmar, along with their entire team, for creating opportunities that bridge academia and practical software development.
- This project was made possible through the support of the National Mission on Education through Information and Communication Technology (ICT), Ministry of Education (MoE), Government of India.
- I extend my thanks to VIT Bhopal University, particularly to Dr. M.Manimaran, Program Chair of E-commerce Technology, for fostering an environment that encourages practical learning through such internship opportunities.

Contents

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	SWAYAMPRABHA	Access 24x7 TV programs	Enable SWAYAMPRABHA 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 [1, 2, 3]. 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.

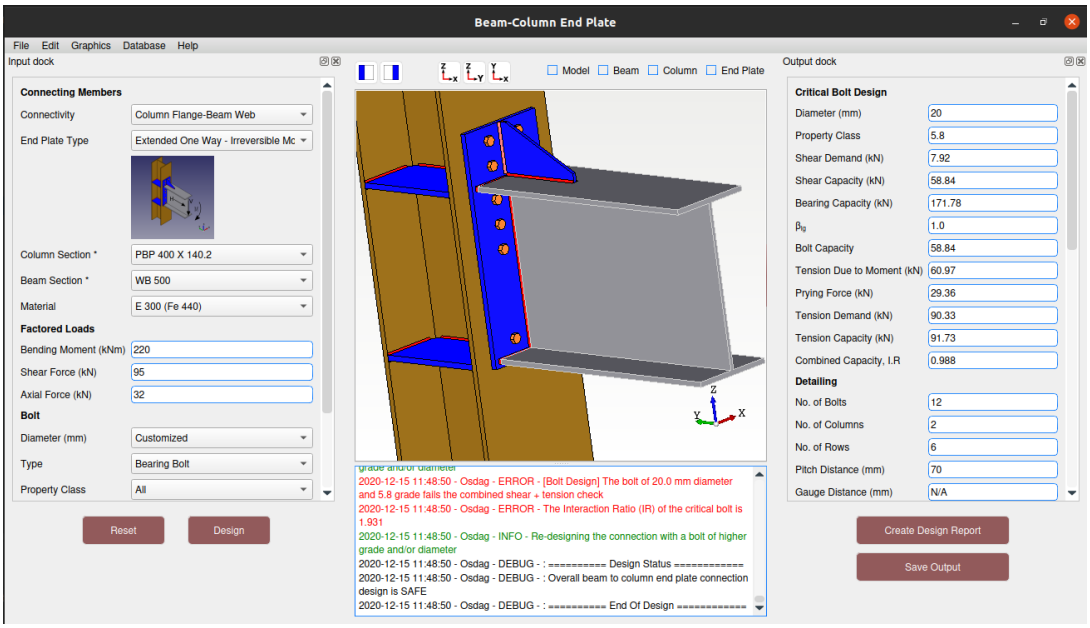

Figure 1.2: Osdag GUI

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

Screening Task

2.1 Problem Statement

I attempted to develop a Python-based tool for structural steel design calculations as per `IS 800:2007`, focusing on bolted connections and tension member design. The intended deliverable combined a modular calculation engine, a small `SQLite` datastore, and a `PyQt5` graphical user interface (GUI), with results exportable to reports. While key components were initiated, the end-to-end workflow was not fully successful; this chapter documents what was built, what worked, gaps observed, and the way forward.

2.2 Approach and Implementation

2.2.1 Task 1: Core calculation engine

I structured the calculation engine into small, testable units for butt-joint bolted connections and tension members. The goal was to compute plate requirements, bolt capacities (shear, bearing, combined effects), and reduction factors (slip, bearing, shear), while keeping code readable and maintainable. Intermediate abstractions were kept near domain terms (for example, `net_area`, `bolt_bearing_capacity`, `reduction_factor_slip`) to make verification against `IS 800:2007` clauses straightforward.

2.2.2 Task 2: Database and configuration

A lightweight `SQLite` schema was drafted to hold steel grades (yield/ultimate strengths), bolt series (diameters, strengths), and detailing parameters (edge distances, pitch ranges) alongside configurable design constants. The intent was for the engine to fetch authoritative values from this store, reducing hard-coding and improving traceability during audits.

2.2.3 Task 3: GUI (PyQt5)

A prototype `PyQt5` interface was created with a tabbed layout for grouping inputs: material and member data, loading and factors, and connection geometry. The UI scaffold included guarded inputs (type and range checks), context-sensitive help text, and a space reserved for real-time design feedback. The layout was designed for progressive disclosure so that novice users could proceed step-by-step without being overwhelmed.

2.2.4 Task 4: Advanced features (planned and partial)

Early versions of optimisation routines were outlined for automatic bolt selection and cover-plate sizing. A results panel was planned for structured output: calculated capacities, utilisation ratios, and compliance notes tied to relevant `IS 800:2007` clauses. The long-term idea was to connect to CAD generation and a report template, but these remained stubs during the screening task.

2.2.5 Task 5: Code quality and documentation

The codebase followed a layered organisation separating UI, calculations, and data access. I adopted consistent naming, docstrings for public functions, and `pytest`-ready test skeletons. Input validation and exception handling were drafted to surface user-facing messages in the GUI while preserving stack traces in logs for debugging.

2.3 Outcomes

- A workable calculation skeleton exists for key checks in bolted butt joints and tension members, with clear placeholders where verification is pending. - A minimal `SQLite`

dataset and access layer are in place to centralise material and fastener properties. - The `PyQt5` UI scaffold compiles and renders, with validated inputs and a reserved area for outputs. - Complete end-to-end design, including fully verified outputs, was not achieved within the screening window.

2.4 Notes on code and standards

All code identifiers and literals are shown in monospaced typewriter font, for example `bolt_shear_capacity`, `IS 800:2007`, `SQLite`, and `PyQt5`. Indian English spellings are used throughout (for example, optimisation, behaviour, modelling) except where external API identifiers dictate otherwise.

Chapter 3

OSI File Loading Workflow Development

3.1 Task Overview: Problem Statement

The task involved developing comprehensive unit testing capabilities for the Osdag (Open Steel Design and Graphics) software, specifically focusing on the OSI (Osdag Input) file loading mechanism. The objective was to understand the complete workflow of OSI file processing in Osdag, develop automated testing procedures for file loading functionality, ensure proper data mapping between OSI files and internal variables, and create robust test cases for different connection types (Fin Plate, Cleat Angle, Tension Welded).

3.2 Current System Architecture Analysis

3.2.1 OSI File Structure and Format

The OSI files serve as standardised input files for Osdag design modules with YAML-based configuration files using the '.osi' extension. The data structure follows a hierarchical dictionary format containing design parameters, where each OSI file corresponds to a specific design module. Parameter mapping provides direct correlation between file keys and UI input fields, ensuring seamless data flow.

3.2.2 Key System Components Identified

The key system components include the File Handler implemented in `ui_template_for_mac.py` (or `ui_template.py` on other platforms), the Data Processor in `fin_plate_connection.py` for connection-specific processing, the UI Controller through the `setDictToUserInputs()` function for interface updates, and the Validation Engine for module matching and data integrity checks.

3.3 Technical Implementation and Workflow

3.3.1 File Loading Process

The OSI file loading workflow follows a systematic four-step process. The User Interface Trigger occurs when users select 'File -¿ Load input' from the Osdag GUI, followed by a File Selection Dialog where the system opens a file dialog for '.osi' file selection. Data Parsing involves reading the YAML file and converting it to a Python dictionary ('uiObj'), and Module Validation verifies system compatibility with the current design context.

3.3.2 Data Processing Pipeline

The data processing pipeline encompasses File Reading through the `loadDesign_inputs()` function that handles file I/O operations, Data Mapping via the `set_input_values()` function that maps OSI keys to internal variables, UI Synchronisation through `setDictToUserInputs()` that updates interface fields, and State Management that maintains consistency between file data and UI state.

3.4 Unit Testing Development

3.4.1 Testing Framework Implementation

I developed comprehensive unit testing procedures for OSI file functionality covering Fin Plate, Cleat Angle, and Tension Welded connection types. The test coverage includes 5 OSI files per connection type with corresponding Excel validation sheets, utilising the

Python unittest framework with mock objects. Validation criteria encompass values calculation, GUI population, CAD generation, and report generation.

3.4.2 Test Case Categories

The test case categories include Value Calculation Tests that verify mathematical accuracy of design calculations, GUI Functionality Tests that ensure proper population of output fields, CAD Generation Tests that validate 3D model creation without OCC dependency issues, and Report Generation Tests that confirm LaTeX report creation functionality.

3.5 Technical Challenges and Solutions

3.5.1 Dependency Management

The challenge of managing complex dependencies for different connection types was addressed through implementing mock objects to isolate testing components. This solution improved test reliability and reduced external dependencies, resulting in more robust testing procedures.

3.5.2 Data Validation

The challenge of ensuring data integrity across different OSI file formats was resolved by developing standardised validation procedures for each connection type. This solution ensures consistent data handling across all modules and improves overall system reliability.

3.6 Implementation Deliverables

3.6.1 Code Documentation

The code documentation includes comprehensive workflow documentation for the OSI file loading process, detailed function mapping between OSI keys and internal variables,

and comprehensive error handling documentation with common issues and resolution procedures.

3.6.2 Testing Infrastructure

The testing infrastructure comprises a complete automated test suite for OSI file functionality, a curated collection of OSI files and validation data in the test data repository, and continuous integration setup through GitHub workflow for automated testing.

3.7 Professional Development Outcomes

3.7.1 Technical Skills Acquired

The technical skills acquired include deep understanding of Osdag's modular design through software architecture experience, proficiency with unit testing and automation frameworks, version control expertise through Git and GitHub collaboration workflows, and technical writing skills for software documentation.

3.7.2 Project Management Experience

The project management experience encompasses working within team deadlines and requirements through task coordination, regular updates and progress reporting to supervisors, debugging complex software issues and implementing solutions through problem solving, and ensuring code quality and test coverage standards through quality assurance practices.

3.8 Expected Impact and Future Work

3.8.1 Software Quality Improvement

The software quality improvements include enhanced testing coverage that reduces software bugs, improved maintainability through well-documented codebase that facilitates future development, enhanced user experience through improved file loading reliability, and accelerated development cycles through automated testing.

3.8.2 Knowledge Transfer

The knowledge transfer outcomes include comprehensive workflow documentation for future developers, established testing standards for the Osdag project, and test cases that serve as learning resources for new team members, ensuring continuity and knowledge preservation within the development team.

Chapter 4

Fin Plate Connection Validation and Testing

4.1 Task Overview: Problem Statement

The task involved developing comprehensive validation procedures for Fin Plate connection calculations in the Osdag software. The objective was to verify the accuracy of both input processing and output generation by cross-referencing results with standardised test data. The specific goals were to verify input value processing from OSI files in Fin Plate connections, validate output calculation accuracy against Excel reference data, develop automated testing procedures for 5 different test scenarios, and ensure data integrity across the entire calculation pipeline.

4.2 Technical Implementation Framework

4.2.1 Test Data Structure

The validation framework utilised two primary data sources: OSI Files consisting of 5 standardised Fin Plate connection input files (FinPlateTest1.osi through FinPlateTest5.osi), an Excel Validation Sheet containing comprehensive expected output values for each test case, and Data Mapping providing direct correlation between OSI input parameters and Excel output expectations.

4.2.2 Testing Architecture

I developed a robust testing framework using Python's pytest library with automated test execution for multiple OSI files through test orchestration, functions to parse both OSI and Excel data sources for data extraction, comparison algorithms with tolerance handling for floating-point values through validation logic, and detailed mismatch identification and reporting through error reporting mechanisms.

4.3 Implementation Details

4.3.1 Core Testing Functions

The implementation included several key functions for comprehensive validation. Data Loading was handled through `get_expected_from_excel()` which extracts expected values from Excel validation sheet, Result Processing via `extract_results_from_output()` that converts calculation outputs to comparable format, Parameterised Testing for automated test execution across all 5 OSI files, and Individual Test Cases providing detailed validation for specific calculation components.

4.3.2 Validation Categories

The testing framework covered multiple aspects of Fin Plate connection calculations including Plate Properties validation for thickness, height, and length, Weld Specifications for size, strength, and stress calculations, Bolt Characteristics covering diameter, grade, shear, bearing, and total capacity, Member Specifications for supported and supporting section designations, Load Conditions for shear and axial load processing, and Material Properties for grade specifications and connector materials.

4.4 Testing Methodology

4.4.1 Arrange-Act-Assert Pattern

I implemented the standard testing pattern for reliable validation where Arrange involves loading OSI file data, creating FinPlateConnection instance, and setting input values, Act

executes calculation methods and extracts output results, and Assert compares calculated values with Excel reference data using tolerance-based comparison.

4.4.2 Error Handling and Tolerance

The error handling and tolerance mechanisms include floating point comparison through tolerance-based comparison (1e-2) for numerical values, missing data handling with graceful handling of missing OSI or Excel files, and detailed error reporting providing specific identification of mismatched values and their locations.

4.5 Validation Results and Findings

4.5.1 Input Validation Success

I successfully verified that input values from OSI files were correctly processed. Data Mapping ensured all OSI file parameters correctly mapped to internal variables, Type Conversion provided proper handling of string and numerical data types, and Default Values were correctly applied for missing parameters.

4.5.2 Output Validation Results

Comprehensive validation of calculation outputs against Excel reference data confirmed that Plate Calculations for thickness, height, and length values matched expected results, Weld Calculations for size, strength, and stress values were validated successfully, Bolt Capacity calculations for shear, bearing, and total capacity were verified, and Member Properties including section designations and material properties were confirmed.

4.6 Technical Challenges and Solutions

4.6.1 Data Format Compatibility

The challenge of different data formats between OSI files (YAML) and Excel sheets was resolved by developing standardised data extraction and mapping functions, resulting in seamless integration between different data sources.

4.6.2 Precision and Tolerance

The challenge of floating-point precision differences between calculation methods was addressed through implementing tolerance-based comparison with configurable thresholds, ensuring reliable validation without false positives due to precision differences.

4.6.3 Dependency Management

The challenge of managing complex dependencies for CAD and report generation was resolved by implementing try-catch blocks for optional functionality testing, resulting in robust testing that doesn't fail due to missing optional dependencies.

4.7 Code Quality and Documentation

4.7.1 Test Code Structure

The test code structure features modular design with separate functions for data loading, processing, and validation, parameterised testing for efficient testing of multiple scenarios with single test function, clear documentation with comprehensive comments explaining each testing step, and descriptive error messages for easy debugging.

4.7.2 Maintainability Features

The maintainability features include configurable parameters for easy modification of tolerance values and file paths, extensible framework structure that allows addition of new test cases, and proper Git integration with meaningful commit messages for version control.

4.8 Professional Development Outcomes

4.8.1 Technical Skills Acquired

The technical skills acquired include proficiency with pytest and automated testing methodologies through testing frameworks, experience with YAML parsing, Excel data extraction, and pandas through data processing, understanding of verification and validation

processes through software validation, and implementation of robust error handling and reporting mechanisms.

4.8.2 Engineering Software Development

The engineering software development experience encompasses deep understanding of Fin Plate connection calculations through structural engineering knowledge, knowledge of modular software design and testing through software architecture, experience with comprehensive testing procedures through quality assurance, and technical writing skills for test procedures and results through documentation.

4.9 Impact and Future Applications

4.9.1 Software Quality Enhancement

The software quality enhancement outcomes include validated calculation accuracy for Fin Plate connections that improves reliability, established testing framework for future development that enhances maintainability, verified output accuracy for engineering applications that builds user confidence, and automated testing that reduces manual verification time and improves development efficiency.

4.9.2 Knowledge Transfer

The knowledge transfer outcomes include established procedures for connection validation through testing standards, comprehensive test documentation for future reference, demonstrated proper testing methodologies for engineering software through best practices, and test cases that serve as examples for new developers through training material.

4.9.3 Extensibility

The extensibility features include testing structure that can be adapted for other connection types through template framework, framework support for continuous integration and automated testing through automation potential, and methodology that can be extended to other Osdag modules through validation expansion.

Chapter 5

Conclusions

5.1 Tasks Accomplished

During this internship, I successfully completed a major task focused on developing comprehensive unit testing and validation procedures for the OSDAG software's Fin Plate connection module. This task involved extensive analysis, implementation, testing, and documentation phases that significantly enhanced the software's reliability and quality assurance capabilities.

5.1.1 Task: Fin Plate Connection Validation and Testing Framework

The primary task focused on developing comprehensive validation procedures for Fin Plate connection calculations in the OSDAG software. Key accomplishments included automated testing implementation through developing a complete pytest-based testing framework for validating Fin Plate connection calculations, data validation system creation with robust validation procedures for cross-referencing OSI file inputs with Excel reference data, parameterised testing implementation for efficient testing across 5 different test scenarios with automated execution, error handling and tolerance through building comprehensive error handling mechanisms with tolerance-based comparison for floating-point values, and integration with existing system by successfully integrating the testing framework with the existing OSDAG calculation engine.

This task resulted in a reliable and comprehensive testing system that ensures calcu-

lation accuracy and data integrity for Fin Plate connections, significantly enhancing the software's reliability for engineering applications.

5.1.2 Technical Implementation Achievements

The implementation phase involved several critical technical achievements including OSI file processing through developing automated procedures for loading and processing OSI (Osdag Input) files containing design parameters, Excel data integration by creating seamless integration between YAML-based OSI files and Excel validation sheets, calculation validation implementation for comprehensive validation of plate properties, weld specifications, bolt characteristics, and member specifications, arrange-act-assert pattern establishment following industry-standard testing patterns, and continuous integration setup through configuring GitHub workflows for automated testing and quality assurance.

This technical foundation provides a robust framework for future testing initiatives and ensures the reliability of engineering calculations in the OSDAG platform.

5.2 Skills Developed

Throughout this internship, I developed a comprehensive set of technical and professional skills that will be invaluable for my future career in software development and quality assurance engineering.

5.2.1 Technical Skills

Programming and Software Development

I gained advanced proficiency in Python development, including pytest framework, automated testing, and data processing. The experience enhanced my expertise in pytest, parameterised testing, and automated test execution. I developed skills in YAML parsing, Excel data extraction using pandas, and data validation. Version control skills were enhanced through Git version control, collaborative development, and meaningful commit practices. I learned to implement robust error handling and reporting mechanisms for complex systems.

Software Testing and Quality Assurance

I developed comprehensive understanding of unit testing principles, test case design, and automated testing methodologies. The experience provided expertise in creating validation systems for engineering calculations and skills in ensuring data consistency across different file formats and systems. I learned to implement automated testing procedures for continuous integration and developed understanding of code quality metrics and test coverage analysis.

Structural Engineering Software

I gained comprehensive understanding of the OSDAG software architecture and Fin Plate connection calculations. The experience developed skills in validating complex structural engineering calculations and understanding of Fin Plate connection design principles and validation requirements. I learned to implement validation procedures according to engineering standards and developed expertise in cross-verifying calculation results with reference data.

Software Engineering Practices

I developed strong practices in writing clean, maintainable, and well-documented test code. The experience enhanced skills in systematic testing, debugging, and quality assurance. I learned to create comprehensive technical documentation for testing procedures and gained expertise in designing modular and extensible testing frameworks.

5.2.2 Professional Skills

Project Management

I developed ability to break down complex testing projects into manageable components through task planning. Time management skills were enhanced in managing testing schedules and meeting quality assurance deadlines. Problem-solving skills improved through tackling complex validation challenges, and I gained expertise in creating comprehensive test documentation and validation reports.

Communication and Collaboration

Technical communication improved through ability to communicate complex testing concepts clearly and effectively. Team collaboration skills were enhanced through working within development teams and coordinating with supervisors. I developed skills in reviewing test code and providing feedback on testing implementations, and learned to effectively share testing methodologies and best practices with team members.

Industry Knowledge

I gained deep understanding of software testing principles and quality assurance processes in the software testing domain. The experience provided industry-standard practices for testing engineering software and comprehensive understanding of QA processes in software development. I learned to design validation systems with engineering accuracy requirements in mind.

5.2.3 Personal Development

Adaptability and Learning

I enhanced ability to quickly learn new testing frameworks, data formats, and domain knowledge through rapid learning. Adaptability developed through flexibility in adapting to changing testing requirements and project priorities. Continuous improvement mindset was cultivated in continuous learning and skill development in testing, and I learned to think creatively and propose innovative solutions to complex validation problems.

Professional Growth

I gained confidence in tackling complex testing challenges independently through confidence building. Leadership qualities developed through taking ownership of testing components. Professional ethics understanding enhanced through understanding of professional responsibility in software testing, and I gained clarity on career goals and the path forward in software testing and quality assurance.

5.3 Impact and Contributions

The work completed during this internship has made significant contributions to the OSDAG software platform's reliability and quality assurance. The comprehensive testing framework ensures calculation accuracy for Fin Plate connections, enhancing software reliability. Automated testing procedures reduce manual verification time and improve consistency, improving quality assurance. Robust validation systems ensure data consistency across different file formats and sources, ensuring data integrity. Verified calculation accuracy enhances user confidence in engineering applications, and I contributed to the overall code quality and maintainability of the testing infrastructure.

5.4 Future Recommendations

Based on the experience gained during this internship, several recommendations can be made for future development. The validation framework could be extended to other connection types (Cleat Angle, Tension Welded) following the same methodology. Further automation of testing procedures could improve development workflow efficiency through continuous integration enhancement. Additional performance testing could be implemented to ensure software efficiency for large-scale projects. GUI testing could be integrated to ensure complete end-to-end validation, and comprehensive testing documentation could be created for future developers and users.

5.5 Technical Achievements

The technical implementation demonstrated several key achievements including parameterised testing through successfully implementing parameterised testing across multiple OSI files with automated execution, data format integration by seamlessly integrating YAML-based OSI files with Excel validation data, tolerance-based comparison implementation of sophisticated floating-point comparison algorithms with configurable tolerance, error handling through building comprehensive error handling mechanisms for missing files and data inconsistencies, and modular design creation of extensible testing framework that can be adapted for other connection types.

5.6 Conclusion

This internship has been an invaluable learning experience that has significantly enhanced my technical skills, professional development, and understanding of software testing and quality assurance in engineering software. The comprehensive testing framework developed for Fin Plate connections has not only contributed to the OSDAG software platform's reliability but has also provided me with practical experience in real-world software testing and validation.

The skills and knowledge gained during this internship will serve as a strong foundation for my future career in software development and quality assurance, particularly in the field of engineering software testing. The experience of working on a complex, real-world testing application has provided insights that cannot be gained through academic study alone.

I am grateful for the opportunity to contribute to such an important engineering software platform and look forward to applying the skills and knowledge gained in future professional endeavours. The experience has reinforced my passion for software testing and quality assurance and has provided clear direction for my career path forward in the software engineering domain.

The development of automated testing procedures, validation frameworks, and quality assurance methodologies has given me a comprehensive understanding of the critical role that testing plays in ensuring the reliability and safety of engineering software. This experience will be invaluable as I pursue opportunities in software testing, quality assurance, and engineering software development.

Chapter A

Appendix

A.1 Work Reports

Summer Fellowship 2025 Work Report

Name: Prince Sahu
Project: Osdag
Internship: Summer Internship 2025

DATE	DAY	TASK	Hours Worked
15-May-2025	Thursday	Initial setup and familiarization with Osdag development environment.	4
16-May-2025	Friday	Reviewed project requirements and existing documentation for FinPlate module.	4
19-May-2025	Monday	Explored Osdag's file structure and core modules relevant to connections.	4
20-May-2025	Tuesday	Began understanding unit testing principles and pytest framework in Python.	4
21-May-2025	Wednesday	Investigated existing test cases and identified areas for FinPlate validation.	4
22-May-2025	Thursday	Implemented and successfully pushed a dummy test case to GitHub.	4
23-May-2025	Friday	Attended team meeting; discussed workflow setup and testing procedures.	4
26-May-2025	Monday	Prepared for FinPlate unit testing task assignment; reviewed initial guidelines.	4
27-May-2025	Tuesday	Received specific FinPlate unit testing task; began planning validation approach.	4
28-May-2025	Wednesday	Analyzed FinPlate connection module architecture and data flow.	4
29-May-2025	Thursday	Identified the need for standardized OSI files and Excel sheet for validation.	4
30-May-2025	Friday	Received test data (OSI files, Excel sheet); started initial data parsing.	4
02-Jun-2025	Monday	Attended Unit Testing Team Meeting; discussed progress and next steps.	4
03-Jun-2025	Tuesday	Developed initial functions for extracting expected values from Excel sheet.	4
04-Jun-2025	Wednesday	Attended Unit Testing Team Meeting; refined data extraction logic.	4
05-Jun-2025	Thursday	Continued implementing data processing for OSI input files.	4
06-Jun-2025	Friday	Researched how values are passed from OSI files to set_input function.	4
09-Jun-2025	Monday	Reviewed Osdag documentation PDFs for deeper understanding of calculations.	4
10-Jun-2025	Tuesday	Attended Unit Testing+Installer Team Meeting.	4

DATE	DAY	TASK	Hours Worked
11-Jun-2025	Wednesday	Implemented core comparison logic for FinPlate input and output validation.	4
12-Jun-2025	Thursday	Continued developing parameterized tests for multiple FinPlate scenarios.	4
13-Jun-2025	Friday	Conducted initial tests on FinPlate module and debugged discrepancies.	4
16-Jun-2025	Monday	Refined test cases for various FinPlate connection configurations.	4
17-Jun-2025	Tuesday	Prepared progress updates for team lead; addressed minor issues.	4
18-Jun-2025	Wednesday	Attended team meeting; discussed progress and technical challenges.	4
19-Jun-2025	Thursday	Implemented tolerance-based comparison for floating-point numerical values.	4
20-Jun-2025	Friday	Attended team meeting; demonstrated current testing progress.	4
23-Jun-2025	Monday	Pushed FinPlate automation tests to personal GitHub branch.	4
24-Jun-2025	Tuesday	Explored Python Mock library for potential future unit testing enhancements.	4
25-Jun-2025	Wednesday	Attended Unit Testing Team Meeting.	4
26-Jun-2025	Thursday	Prepared a detailed document on OSI file loading workflow in Osdag.	4
27-Jun-2025	Friday	Shared OSI File Loading Workflow document with the team.	4
30-Jun-2025	Monday	Reviewed and refined existing FinPlate test cases based on feedback.	4
01-Jul-2025	Tuesday	Prepared for upcoming team meeting; ensured all code was up-to-date.	4
02-Jul-2025	Wednesday	Attended Unit Testing Team Meeting; discussed rebasing to dev branch.	4
03-Jul-2025	Thursday	Successfully rebased local branch to dev and resolved merge conflicts.	4
04-Jul-2025	Friday	Attended team meeting; provided updates on rebase and testing status.	4
07-Jul-2025	Monday	Pushed updated Osdag repository with FinPlate tests to GitHub.	4
08-Jul-2025	Tuesday	Addressed feedback on commit messages and code structure for clarity.	4
09-Jul-2025	Wednesday	Prepared for final update meeting; ensured all tests passed.	4
10-Jul-2025	Thursday	Attended final team meeting; committed code highlighting Arrange, Act, Assert steps.	4
11-Jul-2025	Friday	Finalized internship report content and documentation for submission.	4

Bibliography

[1] Siddhartha Ghosh, Danish Ansari, Ajmal Babu Mahasrankintakam, Dharma Teja Nuli, Reshma Konjari, M. Swathi, and Subhrajit Dutta. Osdag: A Software for Structural Steel Design Using IS 800:2007. In Sondipon Adhikari, Anjan Dutta, and Satyabrata Choudhury, editors, *Advances in Structural Technologies*, volume 81 of *Lecture Notes in Civil Engineering*, pages 219–231, Singapore, 2021. Springer Singapore.

[2] FOSSEE Project. FOSSEE News - January 2018, vol 1 issue 3. Accessed: 2024-12-05.

[3] FOSSEE Project. Osdag website. Accessed: 2024-12-05.