FOSSEE Autumn Semester Long Internship Report

On

Development of Moment Connection Modules for Osdag in Desktop and Web Versions

Submitted by

Navnit Kumar

3rd Year B.Tech Student, Department of CSE

VIT Bhopal University

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

February 3, 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

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.

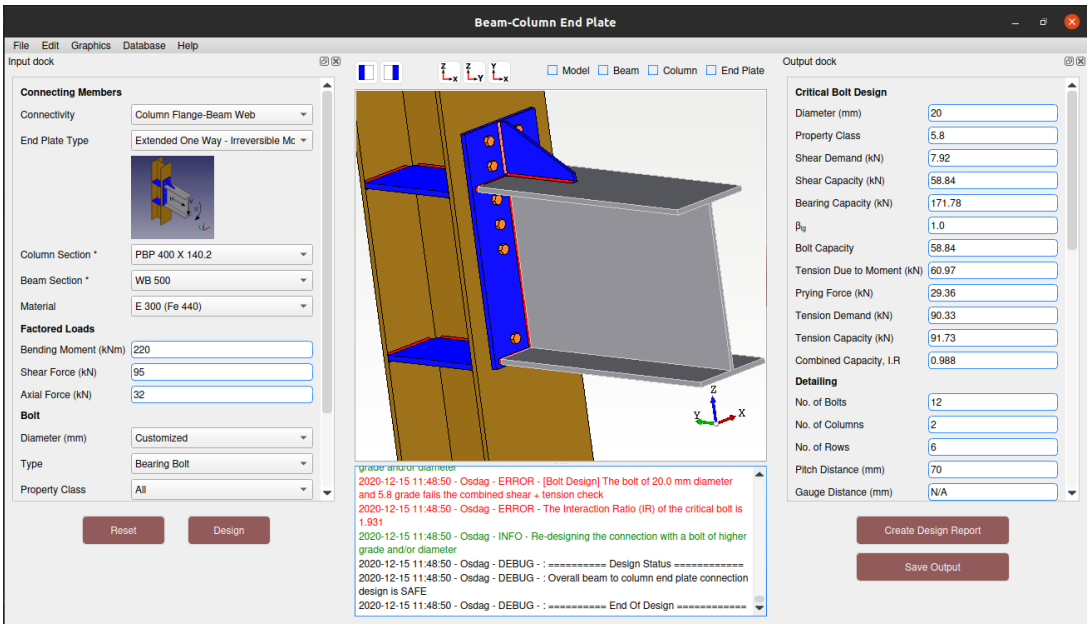


Figure 1.2: Osdag GUI

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

Applicants were tasked to design and develop a modular, extensible, and containerized web-based version of the Prism Viewer application. The system aims to provide interactive three-dimensional visualization of various prism geometries through a modern web interface. The application must support multiple prism types, offer real-time user interaction through configurable controls, and maintain a clean and consistent user interface. Emphasis is placed on adopting a component-based architecture to ensure scalability, maintainability, and ease of future enhancements.

2.2 Tasks Done

The following tasks were performed as part of the screening assignment to develop the Prism Viewer web application:

- **Interactive 3D Visualization:** Implemented real-time 3D rendering of five different prism types triangular, rectangular, pentagonal, hexagonal, and octagonal using Three.js integrated with React.
- **Modular Application Architecture:** Designed a modular and extensible architecture by separating concerns into distinct components for scene rendering, user controls, and data handling.
- **Component-Based Structure:** Developed core components including:

  - `PrismViewer`: Main container component responsible for layout and state management.
  - `PrismScene`: Handles 3D scene setup, rendering, lighting, and camera controls.
  - `ControlPanel`: Provides interactive controls and displays real-time statistics.
  - `types/prism.ts`: Defines TypeScript interfaces and data models for prism configurations.

- **Interactive User Controls:** Added features such as wireframe mode, automatic rotation, orbit controls, and a statistics panel to enhance user interaction with the 3D models.

- **User Interface Design:** Implemented a clean black-and-white theme with a minimalistic design, adhering strictly to the requirement of avoiding gradients.

- **Responsive Design Implementation:** Ensured the application is fully responsive and functions seamlessly across desktop and mobile devices.

- **SEO Optimization:** Incorporated proper meta tags and semantic HTML structure to improve search engine visibility and accessibility.

- **Git Repository:** The complete source code for the Prism Viewer web application developed as part of the screening task is maintained in a public Git repository. The repository contains the full project structure, modular components, configuration files, and implementation details.

  Repository Link: `https://github.com/Navnit-07/fossee`

Chapter 3

Internship Task 1: Refactor Beam to Column End Plate in Osdag Desktop version

3.1 Task 1: Problem Statement

The objective of this task was to refactor the Beam to Column End Plate connection module in the Osdag Desktop version to improve code structure, maintainability, and clarity without altering its functional behavior or design accuracy. The refactoring aimed to align the codebase with best software engineering practices while ensuring continued compliance with relevant design standards. This task was essential to enhance the long-term sustainability of the Osdag codebase and facilitate easier debugging, testing, and future development.

3.2 Task 1: Tasks Done

The following tasks were carried out to address the identified issues and improve the overall functionality of the module:

1. Removal of self-Related Errors and Code Cleanup Several runtime errors caused by incorrect or redundant usage of instance references (self) were identified and resolved across the design and CAD logic files. Duplicate method calls and improper superclass method invocations were corrected to ensure stable execution and improved code consis-

tency.

2. Refactoring of Beam-to-Column End Plate CAD Logic The CAD generation logic for the Beam-to-Column End Plate connection was refactored. Object references were corrected, redundant assignments were removed, and CAD component calls were streamlined to improve clarity and maintainability.

3. Fixing CAD Color Inconsistencies Issues related to incorrect or inconsistent color representation of CAD components such as beams, columns, plates, welds, and bolts were fixed. Proper color mapping was applied to ensure clear visual distinction between different structural elements in the 3D CAD view.

4. Implementation of Hover-Over Functionality for CAD Components Interactive hover functionality was added to CAD components in the Beam-to-Column End Plate module. A hover dictionary was populated with relevant component details (such as bolt grade, diameter, number of bolts, and plate dimensions), enabling informative tooltips when users hover over CAD elements. This significantly improved user understanding and interaction with the 3D model.

5. Improvement of Typical Detailing Popup Interface The user interface for the Typical Detailing popup was improved to enhance clarity and usability. Layout handling and output button connections were corrected to ensure the popup opens correctly and displays relevant detailing information for the Beam-to-Column End Plate connection.

6. Fixing Report Generation Issues The report generation functionality, which was previously not working correctly for this module, was debugged and fixed. Data flow between the design backend and the LaTeX-based report generator was corrected to ensure successful generation of detailed design reports.

7. Enhancement of Module Navigation and UI Integration The Beam-to-Column End Plate connection module was properly integrated into the Osdag GUI navigation. Database configuration and main window logic were updated to ensure the module opens correctly from the home page and loads previous design inputs where available.

8. Normalization and Validation of Input Parameters Input handling for bolt diameter values was improved by implementing normalization logic to handle multiple input formats (e.g., “M16”, “16”, lists of diameters). This enhanced robustness and prevented input-related runtime errors.

9. Logging Improvements and Error Handling Logging statements were standardized

using the Osdag custom logger to ensure consistent warning, error, and information messages during design checks. This improved traceability during debugging and enhanced user feedback during design validation.

10. Testing and Verification After implementing the above changes, the module was tested for multiple design scenarios to verify that CAD visualization, hover functionality, UI behavior, and report generation worked correctly without affecting the design calculations

3.3 Task 1: Python Code

This section presents the key Python code modifications carried out during the internship as part of refactoring and enhancing the Beam to Column End Plate connection module in the Osdag Desktop application.

3.3.1 Description of the Script

The modified Python code is structured into the following functional components:

- **Input Handling and Resource Configuration**: Updates were made to input image references and configuration parameters used in the Beam to Column End Plate module to ensure correct visualization and UI behavior.
- **Design and Output Integration**: The connection design object interacts with Osdag's backend to process design inputs and generate outputs required for CAD views and report generation.
- **CAD Visualization and Interaction**: The CAD logic assigns appropriate colors to connection components such as beams, columns, bolts, and plates. Hover-over functionality was added to CAD elements to display relevant component information interactively.
- **UI and Popup Management**: Python code was refactored to correct UI widget usage and improve the behavior of the typical detailing popup window.
- **Report Generation**: Issues in passing design data to the LaTeX-based report generator were fixed to ensure successful report creation.

3.3.2 Python Code

A representative portion of the modified Python code related to input configuration and report generation is shown below.

Listing 3.1: Beam-to-Column End Plate Module Enhancements

```
# Import required modules
from importlib.resources import files
from osdag_core.Common import *
from osdag_core.latex.CreateLatex import CreateLatex

class BeamColumnEndPlate:
    def input_values(self):
        options_list = []

        # End plate type selection
        options_list.append(
            (KEY_ENDPLATE_TYPE, KEY_DISP_ENDPLATE_TYPE,
             TYPE_COMBOBOX, VALUES_ENDPLATE_TYPE, True, 'No Validator')
        )

        # Updated image reference for correct visualization
        options_list.append(
            (KEY_IMAGE, None, TYPE_IMAGE,
             str(files("osdag_core.data.ResourceFiles.images")
             .joinpath("BC_CF-BW-Flush.png")), True, 'No Validator')
        )

        return options_list

    def save_design(self, popup_summary):
        filename = popup_summary['filename']
        CreateLatex.save_latex(
            CreateLatex(),
            self.report_input,
            self.report_check,
            popup_summary,
            filename,
            self.rel_path,
```

```
34              self.disp_2d_image,
35              self.disp_3d_image,
36              module=self.module
37          )
```

3.3.3 Explanation of the Code

- **Import Statements**: Required Osdag core modules and LaTeX report generation utilities are imported to support UI configuration and report creation.
- **Input Configuration**: The `input_values()` method defines user input options, including the end plate type and associated reference image. Incorrect image paths were corrected to ensure accurate UI representation.
- **Report Generation Logic**: The `save_design()` method handles report generation by passing validated design inputs and outputs to the Osdag LaTeX engine. Errors that previously prevented report generation were fixed by correcting method calls and parameter handling.
- **Integration with CAD and UI**: Additional supporting code (not shown here) handles CAD color assignment, hover-over interaction for connection components, and improved behavior of the typical detailing popup window.

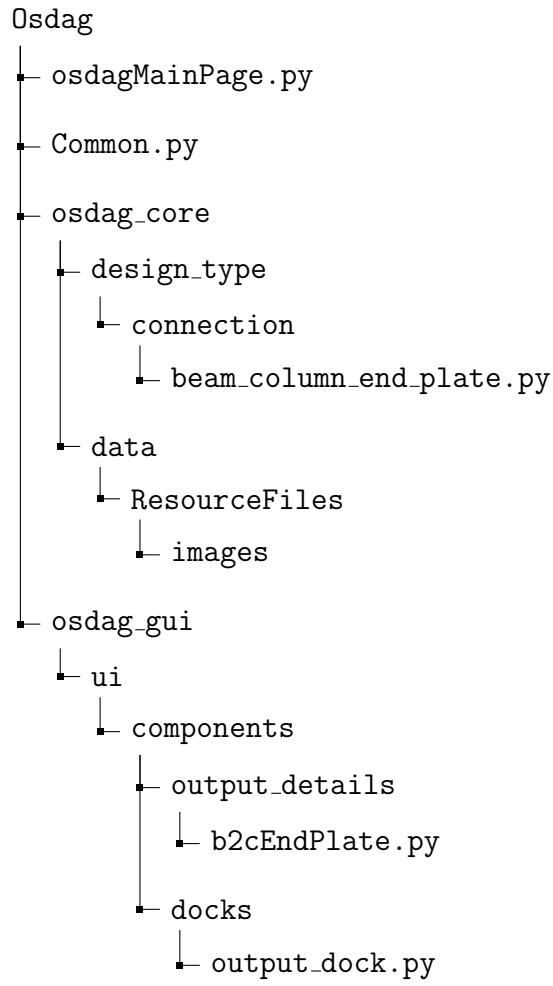
3.4 Task 1: Documentation

This section documents the contributions made towards the Osdag Developer and User Manual as part of Task 1. The documentation focuses on clarifying the structure, workflow, and recent enhancements of the Beam to Column End Plate connection module, particularly in relation to UI behavior, CAD visualization, and report generation.

The updates aim to help future developers and users understand the module organization, typical detailing interface, CAD interaction features, and execution flow within the Osdag Desktop application.

3.4.1 Directory Structure

The Beam to Column End Plate connection module is distributed across multiple directories within the Osdag codebase. The following directory structure highlights the key files and folders relevant to the module and the modifications carried out during the internship.



3.4.2 Beam to Column End Plate Module Workflow

The Beam to Column End Plate module follows a structured workflow:

- User inputs are collected through the GUI, including beam and column sections, end plate type, bolt details, and loading conditions.
- Design calculations are performed using Osdag core design classes in accordance with IS 800:2007.

- CAD detailing is generated using Qt-based graphics components, with improved color schemes and hover-based interaction for better component identification.
- Typical detailing popups display bolt patterns and spacing details using a dialog-based interface.
- Design reports are generated using the LaTeX engine and exported in a user-readable format.

Chapter 4

Internship Task 2: Web-Based Integration of Beam-to-Column End Plate Connection in Osdag

4.1 Task 2: Problem Statement

The Osdag web application did not support the Beam-to-Column End Plate connection module despite the availability of backend design logic. The absence of proper API routes, frontend integration, CAD visualization support, and OSI file compatibility prevented users from performing this design through the web interface. Additionally, issues related to CAD model generation, report generation, UI inconsistencies, and logging affected functionality and user experience. Therefore, it was necessary to integrate, stabilize, and enhance the Beam-to-Column End Plate module in the Osdag web platform.

4.2 Task 2: Tasks Done

This task focused on the complete integration and stabilization of the *Beam-to-Column End Plate Connection* module in the Osdag web application. The work involved coordinated changes across the backend API, CAD generation logic, frontend user interface, and visualization components to ensure feature parity with the desktop version and a seamless user experience.

4.2.1 Backend API Development

- Added a dedicated backend module for the Beam-to-Column End Plate connection to process user inputs received from the web interface.
- Implemented robust input validation and error handling to ensure incorrect or missing parameters are reported clearly to the frontend.
- Developed and refined the output generation pipeline to correctly extract design results, detailing data, stiffener and weld information, and log messages from the core design engine.
- Integrated a custom logging mechanism to capture design warnings, checks, and failures and return them to the web client.

4.2.2 CAD Model Generation and Export

- Extended the CAD model API to support Beam-to-Column End Plate connections, enabling generation of individual and combined component models.
- Fixed CAD section handling by correctly mapping components such as Beam, Column, and Connector for both individual and combined model views.
- Implemented compound CAD model creation to merge multiple parts into a single model for full connection visualization.
- Enabled export of CAD models in multiple formats including BREP, STL, STEP, and IGES for compatibility with external CAD software.
- Added support for per-component STL export to allow selective visualization and improved interaction within the web viewer.
- Fixed CAD color and component recognition issues to ensure correct visual distinction of connection elements.

4.2.3 Frontend Integration and Routing

- Added frontend routing and module registration for the Beam-to-Column End Plate connection to make it accessible from the web application.

- Updated module constants and mappings to ensure consistent communication between frontend routes, backend APIs, and OSI files.
- Integrated the module into the engineering workflow with appropriate camera views and default component selections.

4.2.4 User Interface Enhancements

- Fixed UI inconsistencies related to end plate type selection and image mapping for different beam–column configurations.
- Added and mapped representative images for flushed, extended one-way, and extended both-way end plate configurations.
- Resolved hover flickering issues in module selection cards by refining transition logic and pointer event handling.
- Improved popup behavior and interaction feedback to enhance overall usability.

4.2.5 OSI File Compatibility

- Enhanced OSI file import logic to support multiple formats of module identification, including nested and alternate key names.
- Ensured backward compatibility with OSI files generated by different Osdag modules and versions.

4.2.6 Visualization and Interaction Improvements

- Enhanced the web-based CAD viewer to correctly group and identify connection components.
- Added hover interaction support on CAD components to improve part identification and user interaction.
- Fixed connector rendering issues and ensured consistent visibility across different view selections.

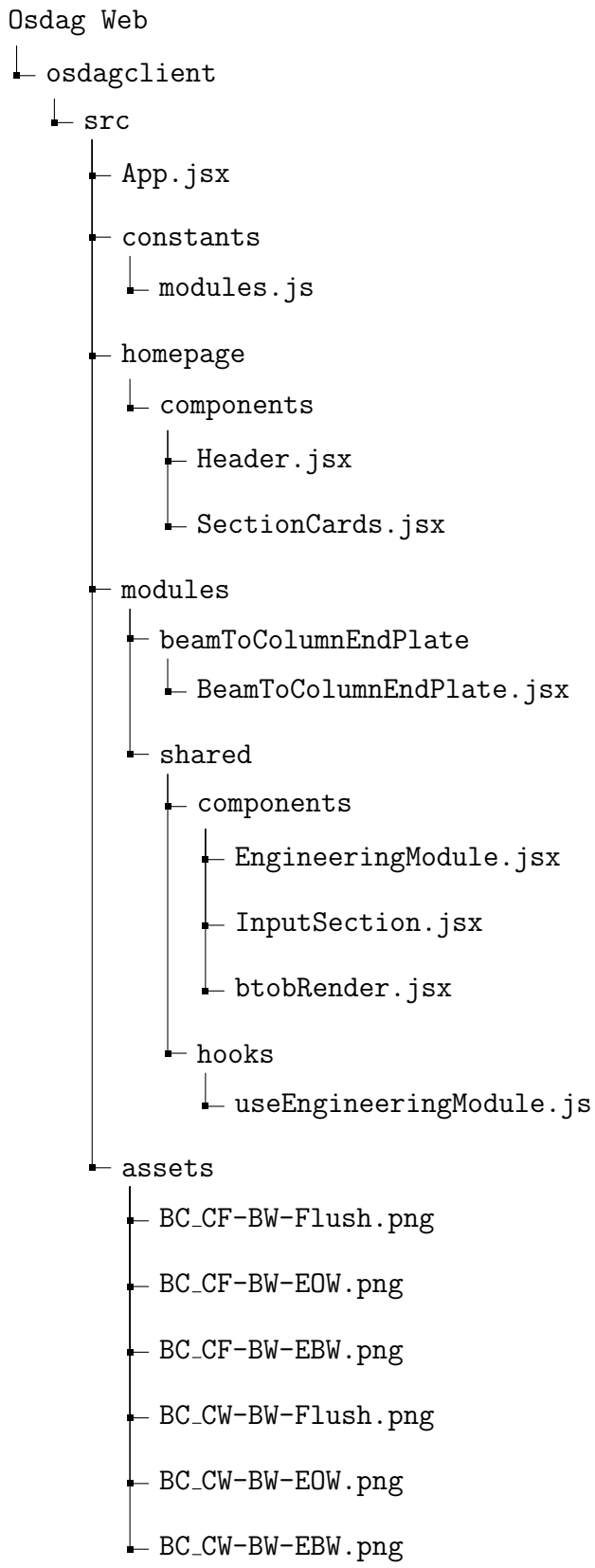
4.2.7 Testing and Validation

- Tested the complete design workflow from input entry to result generation, CAD visualization, and report export.
- Verified correctness of design outputs and CAD models by comparing results with the Osdag desktop version.
- Performed regression testing to ensure changes did not affect existing web modules.

4.3 Task 2: Documentation

This section presents the documentation and structure of the *Beam-to-Column End Plate Connection* module implemented for the Osdag web application. It describes the backend APIs, frontend components, routing, CAD generation, and OSI file integration.

4.3.1 Directory Structure



4.3.2 Program Flow and API Usage

- The main frontend entry for the module is `BeamToColumnEndPlate.jsx`, which handles user input and communicates with the backend API.
- The backend API, implemented in `beam_column_end_plate.py`, processes the input, generates design calculations, and produces CAD models in multiple formats (BREP, STL, STEP, IGES).
- Input from the web UI is validated and mapped to the core Osdag module (`BeamColumnEndPlate`) to ensure accurate design computation.
- Output data includes detailed design parameters, stiffener and weld details, and is returned to the frontend along with log messages for transparency.
- CAD models are generated both per-component and as a compound model for the complete connection. STL, STEP, and IGES formats are exported for visualization or external CAD use.

4.3.3 Frontend Routing and Integration

- The module is registered in `modules.js` with the route `/design/connections/column-beam/end_`
- React Router is used to navigate to the module with optional project ID parameters.
- Default selections for end plate types and connection configurations are set in `useEngineeringModule.js`.
- Images representing different end plate types are mapped and dynamically displayed based on user selection in `InputSection.jsx`.
- Module interactions and CAD visualization are managed via `btobRender.jsx` and shared engineering module components.

4.3.4 OSI File Compatibility

- OSI file import logic supports multiple keys for module identification (`Module`, `module`, `module_id`) for robustness.

- Input data from OSI files is parsed and validated to ensure compatibility with the Beam-to-Column End Plate module.

4.3.5 Visualization and CAD Interaction

- CAD components such as Beam, Column, and Connector are correctly grouped and color-coded in the 3D viewer.
- Hover interactions and selective component visibility have been implemented to enhance usability.
- Compound and per-part CAD models are available for download in multiple formats.

4.3.6 Logging and Error Handling

- Custom logging is implemented in the backend to capture warnings, errors, and informational messages.
- Logs are returned alongside the output to the frontend to facilitate debugging and user feedback.

Chapter 5

Internship Task 3: Design of Beam-to-Beam Connections

5.1 Task 3: Problem Statement

The design of beam-to-beam connections is critical for ensuring safe and efficient load transfer in steel structures. The existing implementation in the OSDAG platform exhibited issues such as inconsistent module integration, incomplete design validation, and inadequate handling of unsafe design conditions. These limitations affected the reliability of design results, report clarity, and CAD model generation. The objective of this task was to implement and integrate beam-to-beam connection modules into the Osdag web platform. This involved developing backend APIs, integrating core design logic, enabling CAD model generation, and providing frontend support to allow users to perform complete design, visualization, and reporting of beam-to-beam connections in a seamless and code-compliant manner.

5.2 Task 3: Tasks Done

This task involved the complete implementation and integration of Beam-to-Beam Connection modules in the Osdag web application. The work required coordinated development across backend APIs, core design logic, CAD model generation, and frontend integration to enable accurate and code-compliant design of beam splice connections.

5.2.1 Backend API Development

- Integrated core design modules with the API layer to process user inputs and generate detailed design outputs and log messages.
- Implemented robust input validation and error handling to ensure invalid or incomplete design inputs are clearly reported.
- Ensured consistent output formatting to support result visualization, report generation, and CAD model creation.

5.2.2 CAD Model Generation and Export

- Extended the CAD model API to support beam-to-beam cover plate and end plate connection geometries.
- Implemented component mapping for Beam, Cover Plate, End Plate, Bolts, and Welds to ensure accurate visualization.
- Enabled compound CAD model generation for complete beam splice connection visualization.
- Supported export of CAD models in multiple formats including BREP, STL, STEP, and IGES.

5.2.3 Frontend Integration and Module Configuration

- Added frontend modules, configurations, and routing for all beam-to-beam connection types.
- Updated module constants and mappings to ensure seamless communication between frontend interfaces and backend APIs.
- Integrated default views, and component visibility controls for improved user interaction.

5.2.4 Visualization and User Interaction Enhancements

- Improved CAD viewer support to correctly identify and render beam splice components.
- Ensured consistent visibility of connectors and plates across different view selections.
- Enhanced user interaction with clear representation of individual and combined connection components.

5.2.5 Testing and Validation

- Tested all beam-to-beam connection modules for various loading and geometric conditions.
- Verified correctness of design results by comparing with the Osdag desktop version.

5.3 Task 3: Documentation

The implementation of the Beam-to-Beam Connection modules (Beam Cover Plate Bolted, Beam Cover Plate Welded, and Beam-to-Beam End Plate) involved modifications and additions across the backend, core design logic, frontend, and CAD integration in the Osdag-web application. The following documentation outlines the directory structure and relevant files created or modified for this task.

5.3.1 Directory Structure

```
Osdag-web/
 backend/
    osdag_web_api/
        coverplatebolted_outputView.py
        cover_plate_weld_output.py
        beamBeamEndPlate_outputView.py
 osdag_core/
    design_type/
```

```
        connection/
            beam_cover_plate.py
            beam_cover_plate_weld.py
            beam_beam_end_plate_splice.py
osdagclient/
   src/
        modules/
            coverPlateBolted/
               configs/
                  coverPlateBoltedConfig.js
               components/
                   UI and visualization components
            coverPlateWelded/
               configs/
                  coverPlateWeldedConfig.js
               components/
                   UI and visualization components
            beamBeamEndPlate/
                configs/
                   beamBeamEndPlateConfig.js
                components/
                    UI and visualization components
cad/
   model_api.py  % CAD model generation and export support
design_report/
   ...
```

5.3.2 Backend API

- Implemented robust input validation, error handling, and response generation for design results, warnings, and failure logs.
- Ensured smooth communication with the frontend, returning JSON-formatted re-

sults compatible with web visualization.

5.3.3 Frontend Components

- Configured module routing, refactoring, camera views, default selections, and interaction behaviors for seamless user experience.
- Mapped frontend inputs to backend API endpoints for accurate data exchange and response handling.

5.3.4 CAD Integration

- Extended CAD generation API to support beam-to-beam connections.
- Enabled 3D visualization of individual components and complete beam splice connections.
- Supported multiple CAD export formats including STL, STEP, IGES, and BREP for compatibility with external CAD tools.

Chapter 6

Internship Task 4: Design and Integration of Column-to-Column Connections

6.1 Task 4: Problem Statement

Column-to-column connections are critical structural elements used to transfer axial loads, bending moments, and shear forces between vertically aligned columns. Accurate design and detailing of these connections are essential to ensure structural continuity, safety, and compliance with design codes such as IS 800:2007.

The Osdag web application previously lacked support for column-to-column splice connections, including cover plate bolted, cover plate welded, and end plate configurations. This limited the ability of users to design and visualize common column splice connections through the web interface.

The objective of this task was to implement and integrate column-to-column connection modules into the Osdag web platform. This involved developing backend APIs, integrating core design logic, enabling CAD model generation, and providing frontend support to allow users to perform complete design, visualization, and reporting of column-to-column connections in a seamless and code-compliant manner.

6.2 Task 4: Tasks Done

This task involved the complete implementation and integration of Column-to-Column Connection modules in the Osdag web application. The work required coordinated development across the backend APIs, core design logic, CAD model generation, and frontend integration to enable accurate and code-compliant design of column splice connections.

6.2.1 Backend API Development

- Implemented dedicated backend API endpoints for Column Cover Plate Bolted, Column Cover Plate Welded, and Column-to-Column End Plate connections.
- Integrated core design modules with the API layer to process user inputs and generate detailed design outputs and log messages.
- Implemented robust input validation and error handling to ensure invalid or incomplete design inputs are clearly reported.
- Ensured consistent output formatting to support result visualization, report generation, and CAD model creation.

6.2.2 Core Design Logic Integration

- Added and integrated core structural design modules for column-to-column connections based on IS 800:2007 provisions.
- Implemented design checks for axial force, bending moment, and shear force transfer through cover plates and end plates.
- Integrated capacity checks for plates, bolts, and welds, including tension, shear, block shear, and combined action where applicable.
- Ensured proper design termination for unsafe configurations with detailed logging of failure reasons.

6.2.3 CAD Model Generation and Export

- Extended the CAD model API to support column-to-column cover plate and end plate connection geometries.
- Implemented component mapping for Column, Cover Plate, End Plate, Bolts, and Welds to ensure accurate visualization.
- Enabled compound CAD model generation for complete connection visualization.
- Supported export of CAD models in multiple formats including BREP, STL, STEP, and IGES.

6.2.4 Frontend Integration and Module Configuration

- Added frontend modules, configurations, and routing for all column-to-column connection types.
- Updated module constants and mappings to ensure seamless communication between frontend interfaces and backend APIs.
- Integrated default views, camera settings, and component visibility controls for improved user interaction.

6.2.5 Visualization and User Interaction Enhancements

- Improved CAD viewer support to correctly identify and render column splice components.
- Ensured consistent visibility of connectors and plates across different view selections.
- Enhanced user interaction with clear representation of individual and combined connection components.

6.2.6 Testing and Validation

- Tested all column-to-column connection modules for various loading and geometric conditions.

- Verified correctness of design results by comparing with the Osdag desktop version.

6.3 Task 4: Documentation

The implementation of the Column-to-Column Connection modules (Column Cover Plate Welded, Column Cover Plate Bolted, and Column-to-Column End Plate) involved modifications and additions across the backend, core design logic, frontend, and CAD integration in the Osdag-web application. The following documentation outlines the directory structure and relevant files created or modified for this task.

6.3.1 Directory Structure

```
Osdag-web/
 backend/
    osdag_web_api/
        column_to_column_coverplatebolted_output.py
        column_cover_plate_welded_output.py
        columncolumnendplate_outputView.py
 osdag_core/
    design_type/
        connection/
            column_cover_plate.py
            column_cover_plate_weld.py
            column_end_plate.py
 osdagclient/
    src/
        modules/
            columnCoverPlateBolted/
               configs/
                  coverPlateBoltedConfig.js
               components/
                   UI and visualization components
            columnCoverPlateWelded/
```

```
            configs/
               coverPlateWeldedConfig.js
            components/
                UI and visualization components
        columnEndPlate/
            configs/
               endPlateConfig.js
            components/
                UI and visualization components
cad/
   model_api.py  # CAD model generation and export support
design_report/
   ...
```

6.3.2 Backend API

- Developed REST endpoints to handle user inputs for all three column-to-column connection types.
- Implemented robust input validation, error handling, and response generation for design results, warnings, and failure logs.
- Ensured smooth communication with the frontend, returning JSON-formatted results compatible with web visualization.

6.3.3 Core Design Modules

- Added structural design logic for bolted and welded column cover plates, and column end plate connections.
- Implemented calculations for plate thickness, weld size, axial and shear capacities, and connection safety checks.
- Integrated the modules with existing Osdag-core APIs for uniform handling across connection types.

6.3.4 Frontend Components

- Created React modules for each connection type including configuration files, input forms, and CAD visualization components.
- Configured module routing, camera views, default selections, and interaction behaviors for seamless user experience.
- Mapped frontend inputs to backend API endpoints for accurate data exchange and response handling.

6.3.5 CAD Integration

- Extended CAD generation API to support new column-to-column connections.
- Enabled 3D visualization of individual components and complete connections.
- Supported multiple CAD export formats including STL, STEP, IGES, and BREP for compatibility with external CAD tools.

6.3.6 Testing and Validation

- Verified the design workflow from input submission to output generation, CAD visualization, and report creation.
- Validated design results and CAD models against the Osdag desktop version for correctness and consistency.

Chapter 7

Conclusions

7.1 Tasks Accomplished

During the course of the fellowship, extensive work was carried out on the design and development of structural steel connections in the Open Steel Design and Graphics (OSDAG) software, covering both the desktop and web-based platforms. The following tasks were successfully accomplished:

- Refactoring and implementation of **beam-to-column end plate connections** in both the desktop and web versions of OSDAG.
- Development of three **beam-to-beam connections**, namely:
  - Beam-to-beam end plate connection,
  - Beam-to-beam cover plate welded connection,
  - Beam-to-beam cover plate bolted connection.
- Development of three **column-to-column connections**, namely:
  - Column-to-column end plate connection,
  - Column-to-column cover plate bolted connection,
  - Column-to-column cover plate welded connection.
- Refactoring and improvement of the **base plate module** in the web version of OSDAG, focusing on enhancing code structure, maintainability, and overall performance.

These tasks contributed to expanding the range of steel connection modules available in OSDAG and improving the robustness and usability of the software.

7.2 Skills Developed

The fellowship provided valuable exposure to both technical and professional skill development. The key skills acquired are summarized below:

- In-depth understanding of **structural steel connection design** as per relevant design codes.
- Practical experience in implementing engineering design logic into **software modules**.
- Hands-on experience with **both desktop and web-based application development**.
- Improved proficiency in **refactoring existing codebases** to enhance clarity, efficiency, and scalability.
- Ability to analyze, debug, and validate complex engineering calculations within a software environment.
- Experience in working with an **open-source engineering software framework**.
- Enhanced skills in **technical documentation** and structured reporting.
- Development of professional skills such as problem-solving, systematic thinking, and collaborative development.

Overall, the fellowship strengthened both domain-specific knowledge in structural engineering and practical software development skills, preparing the foundation for future work in engineering software development and research.

Chapter A

Appendix

A.1 Work Reports

Internship Work Report

Name:	Navnit Kumar
Project:	Osdag
Internship:	FOSSEE Semester Long Internship (Autumn) 2025

DATE	DAY	TASK	Hours Worked
01-Oct-2025	Wednesday	Installed OSDAG for desktop version and explored its features through the walkthrough	4
02-Oct-2025	Thursday	Installed OSDAG for desktop version and explored its features through the walkthrough	4
03-Oct-2025	Friday	Installed OSDAG for web version and explored its features through the walkthrough	5
04-Oct-2025	Saturday	Installed OSDAG for web version and explored its features through the walkthrough	4
06-Oct-2025	Monday	Familiarized myself with Osdag project structure and studied different steel connection modules available in the application	4
07-Oct-2025	Tuesday	Understood Beam-to-Column End Plate connection workflow including input parameters, design checks, and output generation.	5
08-Oct-2025	Wednesday	Explored existing GUI implementation for beam-to-column connections and analyzed limitations in current end plate module	4
09-Oct-2025	Thursday	Reviewed CAD visualization workflow and studied how bolts, plates, beams, and columns are rendered in 3D view	5
10-Oct-2025	Friday	Analyzed bolt, weld, and plate design logic in the core module to understand data flow and dependencies	5
11-Oct-2025	Saturday	Studied input validation mechanisms	4
13-Oct-2025	Monday	Exam Leave	0
14-Oct-2025	Tuesday	Exam Leave	0
15-Oct-2025	Wednesday	Exam Leave	0

16-Oct-2025	Thursday	Worked on issues related to bolt diameter and load input handling	5
17-Oct-2025	Friday	Worked on improving code readability and removing “self” errors in Beam-Column End Plate module	6
18-Oct-2025	Saturday	Integrated improved logging using CustomLogger for better debugging and traceability	6
20-Oct-2025	Monday	Improved CAD component selection and visibility controls in the GUI	6
21-Oct-2025	Tuesday	Fixed hover behaviour for bolts, plates, beams, and columns in 3D CAD view	5
22-Oct-2025	Wednesday	Worked on improving 3D CAD labelling and user interaction for Beam-Column End Plate connection	5
23-Oct-2025	Thursday	Integrated Beam-Column End Plate module properly with output dock actions	4
24-Oct-2025	Friday	Tested design flow for multiple end plate configurations and fixed inconsistencies	7
25-Oct-2025	Saturday	Worked on the OSDAG web application and fixed UI issues, such as the flickering problem on hover	4
27-Oct-2025	Monday	Performed testing for Beam-Column End Plate module and resolved UI issues	6
28-Oct-2025	Tuesday	Cleaned up unused code and improved overall stability of the end plate connection module	4
29-Oct-2025	Wednesday	Refined report generation flow and ensured correct output data is passed to LaTeX reports	6
30-Oct-2025	Thursday	Finalized refactoring changes for Beam-Column End Plate connection	7
31-Oct-2025	Friday	Prepared and reviewed final code changes for Beam-Column End Plate refactoring	6
01-Nov-2025	Saturday	Refactored Beam-Column End Plate module, improving code structure, logging, validation, and CAD integration	5
03-Nov-2025	Monday	Reviewed GUI behaviour for end plate detailing output and identified popup styling issues	6
04-Nov-2025	Tuesday	Worked on improving popup layout and window behaviour	5
05-Nov-2025	Wednesday	Fixed UI inconsistencies in popup dialogs and verified theme compatibility	6

Internship Work Report			
Name:		Navnit Kumar	
Project:		Osdag	
Internship:		FOSSEE Semester Long Internship (Autumn) 2025	
06-Nov-2025	Thursday	Fixed popup styling issues and corrected end-plate type image loading in GUI input panel	7
07-Nov-2025	Friday	Analysed existing web API implementation for beam-to-column end plate connection and identified refactoring requirements	6
08-Nov-2025	Saturday	Refactored backend logic for beam-to-column end plate connection in Osdag-web, including input parsing and validation	6
10-Nov-2025	Monday	Fixed CAD-related issues and logger issues in beam-to-column end plate connection for web	7
11-Nov-2025	Tuesday	Completed and submitted refactor of Beam-to-Column End Plate connection for web, ensuring consistency with Osdag core logic and improving output structure	8
12-Nov-2025	Wednesday	Studied different types of Beam-to-Beam splice connections and reviewed existing implementation in Osdag	7
13-Nov-2025	Thursday	Analyzed requirements and design logic for Beam-to-Beam splice modules in Osdag	6
14-Nov-2025	Friday	Reviewed existing Beam-to-Beam splice code and identified issues and missing functionalities	7
15-Nov-2025	Saturday	Started implementation of Beam-to-Beam splice module by setting up input parameters and design workflow	5
17-Nov-2025	Monday	Exam Leave	0
18-Nov-2025	Tuesday	Exam Leave	0

19-Nov-2025	Wednesday	Worked on design calculations and validation logic for Beam-to-Beam splice connection	4
20-Nov-2025	Thursday	Implemented CAD visualization support for Beam-to-Beam modules	6
21-Nov-2025	Friday	Fixed CAD-related issues and improved component visibility for Beam-to-Beam modules	7
22-Nov-2025	Saturday	Worked on integrating Beam-to-Beam splice module with output dock and report generation	4
24-Nov-2025	Monday	Fixed logical and UI issues across three Beam-to-Beam modules	5
25-Nov-2025	Tuesday	Performed testing for multiple load cases and beam configurations for connections	6
26-Nov-2025	Wednesday	Refined CAD drawings, detailing outputs, and improved stability of modules	7
27-Nov-2025	Thursday	Travel Leave	0
28-Nov-2025	Friday	Improved code readability by standardizing naming conventions and assisted a fellow intern in fixing issues	6
29-Nov-2025	Saturday	Resolved reported bugs and improved overall performance of Beam-to-Beam connections	4
01-Dec-2025	Monday	Final testing and validation of all three Beam-to-Beam modules	7
02-Dec-2025	Tuesday	Completed creation and fixing of three Beam-to-Beam splice modules and submitted final PR	6
03-Dec-2025	Wednesday	Studied Column-to-Column connection types and analyzed design requirements and calculation	7
04-Dec-2025	Thursday	Implemented user interface inputs for Column-to-Column connections in Osdag web	8
05-Dec-2025	Friday	Defined backend logics and endpoints	5
06-Dec-2025	Saturday	Integrated UI with backend for Column-to-Column connections and raised PR	6

Internship Work Report			
Name:		Navnit Kumar	
Project:		Osdag	
Internship:		FOSSEE Semester Long Internship (Autumn) 2025	
08-Dec-2025	Monday	Worked on backend API-level validations for Column-to-Column module input parameters	7
09-Dec-2025	Tuesday	Improved input validation and error handling	6
10-Dec-2025	Wednesday	Fixed UI-related issues and ensured correct data flow between UI and backend APIs Enhanced logging and validation feedback for Column-to-Column connection	7
11-Dec-2025	Thursday	Tested Column-to-Column module for different configurations and validated UI behaviour	4
12-Dec-2025	Friday	Resolved bugs related to UI inputs and backend validation logic	7
13-Dec-2025	Saturday	Retested modules after fixes	4
15-Dec-2025	Monday	Final testing and verification of Column-to-Column UI and backend validation implementation	6
16-Dec-2025	Tuesday	Conducted a walkthrough of Osdag desktop application for new interns, explaining overall architecture and available connection modules	5
17-Dec-2025	Wednesday	Assisted new interns with Osdag development environment setup and resolved configuration-related issues	5
18-Dec-2025	Thursday	Assisted interns in running and testing different connection modules and understanding common errors	4
19-Dec-2025	Friday	Worked on setup and configuration of LCC app	5
20-Dec-2025	Saturday	Familiarized myself with the LCC app structure and studied its functionality and workflow	4

22-Dec-2025	Monday	Reviewed Base Plate backend code in Osdag-web and identified areas for refactoring and cleanup	4
23-Dec-2025	Tuesday	Refactored backend logic for Base Plate module in Osdag-web to improve structure and readability	5
24-Dec-2025	Wednesday	Improved input validation and error handling in Base Plate backend API	5
25-Dec-2025	Thursday	Tested Base Plate backend changes and verified correct behaviour with existing calculation logic	4
26-Dec-2025	Friday	Completed final review of Base Plate backend refactoring	4
27-Dec-2025	Saturday	Organized internship work logs and technical notes	4
29-Dec-2025	Monday	Started drafting final internship report	5
30-Dec-2025	Tuesday	Continued writing and formatting internship report	5
31-Dec-2025	Wednesday	Final report additions	6

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.