FOSSEE Semester Long Internship (2026)

On

Development and Enhancement of OsdagWeb: Structural Module Integration, Multi-Format CAD Export, and Web UI Improvements

Submitted by

Navnit Kumar

3rd Year B.Tech Student, Department of CSE

VIT BHOPAL UNIVERSITY

Madhya Pradesh

Under the Guidance of

Prof. Siddhartha Ghosh

Department of Civil Engineering

Indian Institute of Technology Bombay

Mentors:

Parth Karia

June 19, 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.

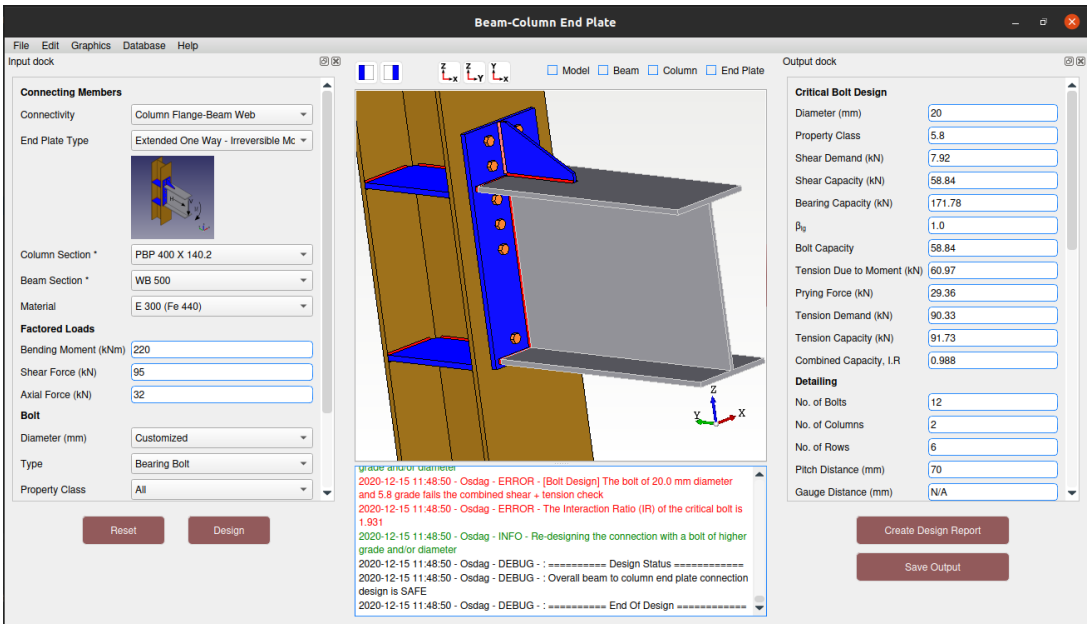

Figure 1.2: Osdag GUI

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

Internship Task 1: Enhancement and Stabilization of Moment Connection Modules in OsdagWeb

2.1 Task 1: Problem Statement

OsdagWeb provides browser-based design workflows for steel moment connections such as beam-to-column end plates, beam-to-beam end plates, cover plate bolted connections, and cover plate welded connections. During the migration from the desktop Osdag application to the web platform, several moment connection modules exhibited inconsistencies in input handling, CAD section mapping, adapter integration, and 3D model rendering.

Existing issues included incorrect CAD part labels (for example, "EndPlate" versus "Connector"), broken end plate rendering, mismatched output configurations, and incomplete backend adapter bindings. These defects prevented reliable design execution and visualization for multiple moment connection types, reducing usability for structural engineers and students.

The objective of this task was to identify, debug, and resolve module-level defects across all moment connection submodules in OsdagWeb, ensuring correct engineering computation, CAD generation, frontend configuration alignment, and end-to-end design workflow stability.

2.2 Task 1: Tasks Done

The following tasks were performed during this internship activity:

- Analysed moment connection module architecture across frontend configs, backend adapters, and `osdag_core` design classes.
- Fixed beam-to-beam end plate module CAD section definitions and connector rendering logic.
- Corrected beam-to-column end plate output configuration and model visualization mappings.
- Resolved cover plate bolted and cover plate welded adapter integration issues.
- Updated CAD API section lists to use consistent part naming conventions (Model, Beam, Connector, CoverPlate).
- Fixed end plate rendering issues in `BBEndplate_cadFile.py` and related common CAD logic.
- Synchronized frontend output image maps and output dock configurations with backend design outputs.
- Validated design report CSV generation paths for updated moment connection modules.
- Tested module workflows through design execution, CAD preview, and output verification.
- Submitted consolidated fixes through Pull Request #37 on the OsdagWeb repository.

2.3 Task 1: Module Architecture Analysis

Moment connection modules in OsdagWeb follow a layered architecture:

- **Frontend Layer:** Module-specific React components and configuration files define input fields, validation rules, output tables, and CAD options.

- **Backend Adapter Layer:** Django adapters translate web input payloads into structures expected by the Python engineering core.
- **Engineering Core:** The `osdag_core` package performs IS 800 based design calculations and invokes CAD model builders.
- **CAD Layer:** OpenCascade-based geometry is generated as BREP models and exposed to the web viewer through section-wise CAD paths.

During analysis, inconsistencies were found between frontend module keys, backend session types, and CAD section identifiers. Harmonizing these identifiers was essential to restore reliable end-to-end behaviour.

2.4 Task 1: Beam-to-Beam and Beam-to-Column End Plate Fixes

A major contribution involved correcting end plate connection modules.

- Updated CAD model API section mappings so beam-beam end plate sessions expose `Model`, `Beam`, and `Connector` sections instead of deprecated labels.
- Fixed model and end plate rendering issues in the CAD generation pipeline, improving 3D visualization fidelity.
- Aligned frontend output configurations (`beamBeamEndPlateOutputConfig.js`, `beamToColumnEndP` with backend result schemas.
- Updated design type modules in `osdag_core/design_type/connection/` for beam end plate and column end plate workflows.

These changes restored predictable CAD part visibility and improved design output presentation in the Output Dock.

2.5 Task 1: Cover Plate Connection Module Fixes

Cover plate bolted and welded modules required adapter and configuration updates to support correct parameter submission and CAD regeneration.

- Corrected welded and bolted cover plate adapter bindings in the backend moment connection submodule tree.
- Updated frontend configuration files for cover plate welded modules, including input defaults and output field mappings.
- Improved CAD helper utilities to support consistent geometry extraction across connection types.
- Verified that design execution produces valid model paths for all cover plate variants.

2.6 Task 1: Integration and Validation

Integration testing was performed to validate interaction between Input Dock, design API, CAD scene manager, and Output Dock.

- Confirmed that updated modules execute design calculations without adapter exceptions.
- Verified CAD scene context updates when switching between Model, Beam, and Connector views.
- Checked output image generation mappings for report and visualization consistency.
- Ensured shared `EngineeringModule` component correctly consumes updated module configurations.

The moment connection fixes formed the first major deliverable within Pull Request #37, contributing approximately 55 changed files and more than 2400 lines of additions across frontend, backend, and core engineering modules.

Chapter 3

Internship Task 2: Integration and Bug Resolution for Tension Member Design Modules

3.1 Task 2: Problem Statement

Tension member design is a core capability of OsdagWeb, supporting bolted and welded connection types for axial tension members. After the web platform revamp, both tension member submodules exhibited defects in input serialization, backend adapter communication, CAD model generation, and output rendering.

Users encountered failures during design submission, incomplete output tables, and inconsistent CAD previews for bolted-to-end and welded-to-end configurations. Because tension member modules share the common `EngineeringModule` workspace framework, unresolved defects also affected overall platform reliability.

The objective of this task was to stabilize both tension member modules (bolted and welded), align frontend and backend contracts, and ensure complete design-to-CAD-to-output workflows for all tension member use cases.

3.2 Task 2: Tasks Done

The following tasks were carried out during this activity:

- Reviewed tension member module structure in frontend configs and backend adapters.
- Fixed bolted tension member adapter parameter mapping and validation issues.
- Fixed welded tension member adapter integration and service bindings.
- Updated `boltedToEndConfig.js` and `weldedToEndConfig.js` for correct input field definitions.
- Corrected output dock configurations for bolted and welded tension member result tables.
- Updated `osdag_core/design_type/tension_member/` modules for web-compatible execution paths.
- Synchronized API route constants and design keys with module identifiers.
- Verified CAD section visibility and model path generation for tension member designs.
- Performed regression testing alongside moment connection fixes in Pull Request #37.

3.3 Task 2: Backend Adapter Corrections

Backend adapters serve as the bridge between React frontend payloads and legacy Python design classes. For tension members, adapter defects caused parameter loss and incorrect design state initialization.

Key backend improvements included:

- Correcting bolted adapter submission mapping in `tension_member/submodules/bolted/adapter`
- Fixing welded adapter logic in `tension_member/submodules/welded/adapter.py`.
- Ensuring module views expose consistent endpoints for design and CAD generation.
- Validating that adapter outputs include complete CAD model path dictionaries for frontend rendering.

These corrections enabled reliable server-side design execution without manual intervention or fallback errors.

3.4 Task 2: Frontend Configuration Alignment

Frontend module configs define the user-facing design experience, including input groups, material and section selectors, bolt configuration panels, and output formatting.

Updates were made to:

- Input configuration files for bolted-to-end and welded-to-end modules.
- Output configuration files specifying result tables, utilization ratios, and failure modes.
- Shared constants (`DesignKeys.js`, `apiRoutes.js`) to maintain consistent module routing.
- Integration with the shared engineering workspace for design, reset, and report generation actions.

Aligning frontend configs with backend schemas eliminated mismatches that previously caused empty outputs or incorrect field labels.

3.5 Task 2: Engineering Core Updates

The Python engineering core modules `tension_bolted.py` and `tension_welded.py` were updated to support web-initiated design runs.

- Ensured compatibility with web adapter parameter formats.
- Preserved IS 800 based design checks and capacity calculations.
- Maintained naming conventions required by CAD generators and report exporters.
- Improved error propagation for invalid input combinations.

3.6 Task 2: Workflow Validation

End-to-end validation was performed for both tension member types:

- User enters project and member parameters in the Input Dock.

- Frontend submits normalized input values to the design API.
- Backend adapter invokes the appropriate tension member design class.
- CAD model paths are returned and rendered in the 3D viewer.
- Output Dock displays design results and utilization summaries.
- Logs capture design steps for debugging and report generation.

Successful validation of both bolted and welded modules confirmed that tension member workflows were production-ready within the OsdagWeb revamp branch.

Chapter 4

Internship Task 3: Development and Integration of Flexural Member Modules

4.1 Task 3: Problem Statement

Flexural member design modules in OsdagWeb support critical structural workflows including simply supported beams, cantilever beams, and purlins. These modules extend the platform beyond connection design into member-level flexure checks and CAD visualization.

During the web migration, the simply supported beam module required bug fixes, while cantilever and purlin modules needed new submodule integration on both frontend and backend. Missing or incomplete implementations prevented users from executing full flexural design workflows in the browser.

The objective of this task was to fix the simply supported beam module, implement cantilever and purlin modules end-to-end, and integrate them with the shared OsdagWeb engineering workspace, CAD viewer, and output/report systems.

4.2 Task 3: Tasks Done

The following tasks were performed during this internship activity:

- Fixed simply supported beam adapter and frontend configuration defects.
- Implemented backend submodule structure for on-cantilever flexural design.
- Implemented backend submodule structure for purlin flexural design.
- Created React workspace components for `OnCantilever.jsx` and `Purlin.jsx`.
- Authored module configuration and output configuration files for all three flexural modules.
- Updated `osdag_core/design_type/flexural_member/` classes for web compatibility.
- Integrated flexural modules with shared CAD scene manager and view mappings.
- Registered module routes and design keys in frontend constants.
- Validated design execution, CAD preview, and output generation for each module.
- Delivered all flexural member changes as part of Pull Request #37.

4.3 Task 3: Simply Supported Beam Module Fix

The simply supported beam module required corrections across adapter logic and frontend configs to restore reliable design execution.

- Fixed parameter mapping in `simply_supported_beam/adapter.py`.
- Updated `SimplySupportedBeam.jsx` integration with the shared engineering hook system.
- Corrected input and output configuration files for beam geometry, loading, support conditions, and design results.
- Verified CAD model generation and section visibility in the 3D workspace.

These fixes enabled users to perform simply supported beam design with consistent input validation and output presentation.

4.4 Task 3: Cantilever Beam Module Implementation

A new cantilever flexural member submodule was added to OsdagWeb with complete frontend and backend integration.

4.4.1 Backend Implementation

- Created `on_cantilever` submodule with adapter, service, and package initialization files.
- Connected adapter to `flexure_cantilever.py` in the engineering core.
- Exposed design and CAD endpoints through flexure member views.

4.4.2 Frontend Implementation

- Developed `OnCantilever.jsx` workspace component.
- Authored `onCantileverConfig.js` and `onCantileverOutputConfig.js`.
- Integrated module with shared Input Dock, Output Dock, Log Window, and CAD viewer components.

The cantilever module supports cantilever beam design workflows with real-time parameter updates and CAD visualization.

4.5 Task 3: Purlin Module Implementation

The purlin module was similarly implemented as a new flexural member submodule.

4.5.1 Backend Implementation

- Created `purlin` submodule with adapter and service layers.
- Integrated with `flexure_purlin.py` design logic in `osdag_core`.

- Ensured CAD model paths are returned in the expected format for frontend scene rendering.

4.5.2 Frontend Implementation

- Developed `Purlin.jsx` workspace component.
- Authored `purlinConfig.js` and `purlinOutputConfig.js`.
- Configured module-specific CAD options and output tables for purlin design results.

4.6 Task 3: Shared CAD and Workspace Integration

Flexural modules rely on shared CAD infrastructure for 3D visualization.

Updates were made to:

- `SceneManager.jsx` for improved scene loading and part selection behaviour.
- `partConfig.js` and `viewMappings.js` for flexural member CAD views.
- `CadSceneContext.jsx` for consistent state management across module switches.
- `flexure.py` base class for common flexural member behaviour.

4.7 Task 3: Module Workflow Summary

The flexural member workflow follows a standardized pattern across all three modules:

- User selects a flexural module from the OsdagWeb home workspace.
- Input parameters are configured through the Input Dock.
- Design API executes IS 800 based flexural checks via backend adapters.
- CAD geometry is generated and displayed in the 3D viewer.
- Results appear in the Output Dock with utilization and design summary tables.
- Logs and reports can be exported through navbar actions.

This task significantly expanded OsdagWeb's member design capabilities and completed a major portion of the platform revamp effort.

Chapter 5

Internship Task 4: Multi-Format CAD Model Export Pipeline for OsdagWeb

5.1 Task 4: Problem Statement

OsdagWeb generates 3D CAD models as part of structural design workflows, but users previously had limited options for exporting models in industry-standard formats. The desktop Osdag application supports saving models in multiple CAD and BIM formats; the web platform required equivalent functionality accessible from the navbar "Save 3D Model" menu.

The existing web implementation supported only basic STL section downloads and lacked a unified export workflow for BREP, STEP, IGES, and IFC formats. Without these exports, users could not easily transfer OsdagWeb designs into external CAD tools, analysis software, or BIM platforms.

The objective of this task was to implement a complete on-demand CAD export pipeline, expose export options through the navbar user interface, and ensure reliable backend regeneration of models from current input values in all supported formats.

5.2 Task 4: Tasks Done

The following tasks were carried out during this activity:

- Extended the backend `exportCad` API endpoint to support BREP, STL, STEP, IGES, and IFC formats.

- Implemented frontend navbar submenu options under "Save 3D Model" for all export formats.
- Developed `exportCad` client integration in the engineering service layer.
- Created cached-model download utilities for BREP and STL when pre-generated models exist.
- Implemented on-demand export regeneration using module input values for STEP, IGES, and IFC.
- Built IFC export pipeline using IfcOpenShell with mesh tessellation from OpenCascade shapes.
- Added mesh coarsening logic to prevent IFC export failures on high-complexity geometries.
- Fixed BREP file reading compatibility with updated OpenCascade Python bindings.
- Updated `requirements.txt` to include IfcOpenShell dependency.
- Submitted and merged changes through Pull Request #40 on the OsdagWeb repository.

5.3 Task 4: Backend Export Architecture

The CAD export system follows an on-demand regeneration model. Instead of relying solely on previously saved files, the backend reconstructs the merged CAD shape from current `input_values` using the module adapter and exports the requested format.

5.3.1 Export API Design

The `CADExport` view accepts POST requests with:

- `module_id` – target design module identifier
- `input_values` – current design parameters

- `section` – CAD section (default: Model)
- `format` – one of `brep`, `stl`, `step`, `iges`, or `ifc`

The API validates inputs, resolves the module adapter, generates the BREP shape, and converts it to the requested export format before returning a binary file response.

5.3.2 Format Conversion Utilities

Export utilities in `cad_export.py` provide:

- `export_step` – STEP file generation from OpenCascade shapes
- `export_iges` – IGES file generation
- `export_ifc` – IFC4 export via IfcOpenShell mesh representation

For IFC export, the pipeline tessellates the shape to STL, parses the mesh, and constructs an IFC4 model with spatial hierarchy (Project, Site, Building, Storey, BuildingElementProxy).

5.4 Task 4: IFC Export Implementation

IFC export presented unique challenges due to mesh complexity limits in IfcOpenShell.

Key implementation details:

- Tessellation uses progressive linear deflection values (1.0 to 32.0) to reduce triangle count.
- A maximum triangle threshold (400,000) prevents worker crashes on complex finplate geometries.
- Binary STL writing is enforced for consistent mesh parsing.
- Temporary STL files are cleaned up after IFC generation.
- Meaningful error messages guide users when mesh coarsening is insufficient.

This approach balances export reliability with acceptable geometric fidelity for BIM interoperability.

5.5 Task 4: Frontend Integration

Frontend changes connected the navbar UI to the export API through the shared engineering service.

5.5.1 Navbar Menu Enhancement

The "Save 3D Model" menu item was extended with submenu options:

- Export BREP
- Export STL
- Export STEP
- Export IGS
- Export IFC

5.5.2 Export Workflow Logic

The `UnifiedDropdownMenu` component implements intelligent export routing:

- For BREP and STL, attempt download from cached CAD model paths if available.
- If cached models are unavailable, fall back to on-demand `exportCad` API regeneration.
- For STEP, IGES, and IFC, always regenerate from current input values via the export API.
- Module submission parameters are built using each module's `buildSubmissionParams` function.
- Downloaded files use Content-Disposition filenames when provided by the backend.

Supporting utilities in `cadExport.js` handle base64 decoding, cached model resolution, and browser download triggering.

5.6 Task 4: Engineering Service Integration

The `useEngineeringService` hook was extended with an `exportCADModel` method, and the `EngineeringModule` component passes `moduleConfig` and `extraState` to the dropdown menu for correct parameter serialization.

This ensures export actions use the same input normalization pipeline as design submission, maintaining consistency between displayed models and exported files.

5.7 Task 4: Testing and Validation

Export functionality was validated across multiple module types:

- Verified BREP and STL downloads from cached design outputs.
- Confirmed STEP and IGES regeneration for modules with valid CAD adapters.
- Tested IFC export on representative connection geometries.
- Validated error handling for missing module IDs and failed tessellation.
- Confirmed successful merge of Pull Request #40 into the `25april` branch.

This task delivered production-ready multi-format CAD export capability, significantly improving OsdagWeb interoperability with external engineering tools.

Chapter 6

Internship Task 5: Navigation Bar, Graphics Controls, and Workspace UI Enhancements

6.1 Task 5: Problem Statement

The OsdagWeb engineering workspace provides a unified interface for all design modules through a shared navbar, Input Dock, Output Dock, Log Window, and 3D CAD viewer. As multiple modules were integrated and export features added, several navbar and graphics control issues emerged affecting usability and workflow consistency.

Reported problems included broken graphics option handling (view toggles and part visibility), incorrect camera view mappings for front/top/side views, disabled design report creation menu action, invalid module ID usage during OSI file save, and inconsistent CAD part selection behaviour when switching between Model, Beam, Column, and connector components.

The objective of this task was to refine navbar menu behaviour, restore graphics and view controls, improve CAD part selection logic, and align workspace actions with expected desktop Osdag workflows.

6.2 Task 5: Tasks Done

The following tasks were performed during this internship activity:

- Fixed navbar graphics option routing for view and part visibility controls.
- Corrected camera view mappings for front, side, and top view actions.
- Restored "Create Design Report" menu action functionality.
- Fixed OSI save workflow to use backend-valid module identifiers.
- Improved CAD part selection to allow exactly one active part at a time.
- Separated background colour picker from generic graphics option handling.
- Updated keyboard shortcuts to match corrected camera view mappings.
- Exposed `setCreateDesignReportBool` through engineering module hooks.
- Refined Quit action to navigate to home page instead of abrupt session termination.
- Submitted UI fixes through Pull Request #87 on the OsdagWeb repository.

6.3 Task 5: Navbar Menu Logic Refactoring

The `UnifiedDropdownMenu` component serves as the central handler for all navbar actions including File, Edit, Graphics, and Database menus.

6.3.1 Graphics Option Handling

Previously, several graphics menu items were routed incorrectly or ignored. The refactored handler introduces explicit cases for:

- **View controls:** Show front view, Show side view, Show top view
- **Part visibility:** Model, Beam, Column, Seated Angle, Cleat Angle
- **Background:** Change Background (via dedicated colour picker trigger)
- **Session:** Quit (navigate to home)

Graphics options now update `inputs.graphicsOption`, which the `EngineeringModule` component processes to adjust CAD camera views and visible sections.

6.3.2 Design Report and OSI Save Fixes

- Re-enabled the "Create Design Report" menu action by restoring `setCreateDesignReportBool(t`
- Fixed OSI save to use `moduleConfig.designType` instead of display names that fail backend validation.
- Improved project name resolution using session name fallbacks.

These fixes prevent silent failures during common file and report operations.

6.4 Task 5: CAD View and Camera Corrections

Camera view mappings were corrected to align navbar actions with the 3D viewer coordinate system.

- "Show front view" now sets camera view to `ZX` (previously incorrect `XY`).
- "Show top view" now sets camera view to `XY` (previously incorrect `ZX`).
- "Show side view" continues to use `YZ`.
- Keyboard shortcuts in `useEngineeringShortcuts.js` were updated to match.

These corrections ensure intuitive view navigation consistent with engineering visualization conventions.

6.5 Task 5: CAD Part Selection Improvements

CAD part selection logic was simplified to improve clarity when toggling visible components.

- Selecting a non-Model part (Beam, Column, Seated Angle, Cleat Angle) now sets exactly one active section.
- Part key normalization maps display labels (e.g., "Seated Angle") to internal keys (e.g., `SeatedAngle`).
- Checkbox interactions in the CAD panel follow the same single-selection model.

- Camera view updates immediately when a new part is selected.

This prevents confusing multi-part overlays and matches expected behaviour from desktop Osdag workflows.

6.6 Task 5: Background Colour Picker Integration

Background colour customization was decoupled from the generic graphics option pipeline.

- A hidden colour input element was added to `EngineeringModule.jsx`.
- `triggerBackgroundColorPicker` callback opens the native colour picker synchronously.
- The dropdown menu invokes this callback for "Change Background" actions.
- Selected colours update the CAD viewer background through `customBgColor` state.

This approach avoids race conditions and ensures reliable background updates.

6.7 Task 5: Hook and State Management Updates

Supporting hook updates ensure navbar actions propagate correctly through the engineering workspace state tree.

- `useDesignReport.js` exports `setCreateDesignReportBool`.
- `useEngineeringModule.js` forwards report state setters to child components.
- `EngineeringModule.jsx` passes `triggerBackgroundColorPicker` to the dropdown menu.
- Graphics option state is cleared after processing to allow repeated selection of the same option.

6.8 Task 5: Review Outcomes and Learnings

Pull Request #87 underwent mentor review, which highlighted important product decisions:

- Background colour change features may be discontinued in favour of a standardized viewer theme.
- Certain legacy view labels (XY, XX) are no longer supported in the revamped CAD system.
- Navbar changes must align with the product roadmap rather than desktop parity alone.

Although the pull request was ultimately closed pending product alignment, the implementation work provided valuable experience in UI regression analysis, mentor feedback incorporation, and iterative refinement of workspace controls.

This task reinforced the importance of validating UI changes against both technical correctness and product direction in collaborative open-source projects.

Chapter 7

Conclusions

7.1 Tasks Accomplished

During the course of the semester-long internship from January to April 2026, significant contributions were made towards the development and stabilization of the OsdagWeb platform under the FOSSEE initiative at IIT Bombay. The work focused on structural design module integration, CAD visualization improvements, multi-format 3D model export, and user interface enhancements for a browser-based steel design environment.

One of the primary contributions was the comprehensive fixing and extension of moment connection modules in OsdagWeb. Multiple connection types including beam-to-beam end plate, beam-to-column end plate, cover plate bolted, and cover plate welded modules were debugged and aligned across frontend configurations, backend adapters, and the `osdag_core` engineering layer. CAD section mappings, end plate rendering, and output dock presentations were corrected to restore reliable design-to-visualization workflows.

A major portion of the internship involved stabilizing tension member design modules for both bolted and welded configurations. Adapter parameter mapping, output table configurations, and engineering core compatibility were resolved, enabling complete axial tension member design execution in the web platform.

Considerable effort was devoted to flexural member module development, including fixes to the simply supported beam module and new implementations for cantilever and purlin modules. These modules were integrated end-to-end with backend services, React workspace components, CAD scene management, and output/report systems, signifi-

cantly expanding OsdagWeb's member design capabilities.

Another important contribution was the development of a multi-format CAD model export pipeline. Export options for BREP, STL, STEP, IGES, and IFC formats were added under the navbar "Save 3D Model" menu, with backend on-demand regeneration and frontend intelligent caching logic. The IFC export implementation using IfcOpenShell with adaptive mesh coarsening addressed practical BIM interoperability requirements.

Additional work was performed on navbar and workspace UI improvements, including graphics control routing, camera view corrections, design report menu restoration, OSI save fixes, and CAD part selection refinements. This work improved overall usability of the shared engineering workspace across all design modules.

7.2 Skills Developed

The internship provided valuable exposure to full-stack web development, structural engineering software, CAD processing pipelines, and collaborative open-source development practices.

Strong proficiency was developed in React.js frontend engineering through work on shared workspace components, module configurations, navbar menus, hooks, and state management patterns. Experience was gained in building reusable UI frameworks that support multiple engineering modules through configuration-driven design.

Significant practical experience was acquired in Django REST Framework backend development, including adapter design, API endpoint implementation, file export services, and integration with legacy Python engineering modules. Understanding of how web APIs wrap desktop-era design logic proved essential for maintaining calculation correctness while enabling browser-based workflows.

The internship provided substantial exposure to CAD and 3D visualization concepts. Working with OpenCascade BREP geometry, mesh tessellation, multi-format export (STEP, IGES, STL, IFC), and frontend CAD scene management enhanced understanding of geometry representation, coordinate systems, and interactive 3D visualization in engineering applications.

Another important learning outcome was BIM interoperability through IFC export development. Implementing IfcOpenShell-based mesh pipelines, managing tessellation

complexity limits, and handling export error cases strengthened skills in practical CAD-to-BIM conversion workflows.

Beyond technical implementation, the internship strengthened abilities in debugging, code review, documentation, version control, and collaborative software development. Regular interaction with mentors through pull request reviews improved communication skills, professional code quality awareness, and understanding of product-driven development decisions in open-source engineering software.

7.3 Future Scope

The work completed during this internship establishes a foundation for continued OsdagWeb development in several directions:

- Further module migration from desktop Osdag to the web platform with consistent adapter patterns.
- Enhanced CAD viewer features including measurement tools, annotation, and improved performance for large models.
- Expanded export formats and cloud storage integration for project and model management.
- Automated regression testing for design modules and CAD export pipelines.
- User authentication, project persistence, and collaborative design workflows.

7.4 Overall Experience

Overall, the FOSSEE Semester Long Internship on OsdagWeb was a highly rewarding learning experience that significantly enhanced software engineering, structural design software development, CAD processing, and open-source collaboration skills. The knowledge and experience gained through this internship provide a strong foundation for future work in full-stack engineering applications, CAD systems, and large-scale technology development.

The contributions made during January–April 2026 directly improved the reliability, functionality, and usability of OsdagWeb, supporting FOSSEE's mission to promote open-source tools for engineering education and professional practice in India.

Chapter A

Appendix

A.1 Internship Work Reports

This section contains the detailed day-wise work reports maintained throughout the internship period from January to May 2026. The reports provide a chronological record of development activities, module integration, CAD export implementation, user interface development, architectural planning, testing, documentation, and integration work carried out during the development of the OsdagWeb platform.

The work reports include information regarding:

- OsdagWeb codebase exploration and local development environment setup.
- Moment connection module fixes and CAD rendering improvements.
- Tension member and flexural member module integration and stabilization.
- Multi-format 3D CAD model export (BREP, STL, STEP, IGES, IFC).
- Navbar, graphics controls, and engineering workspace UI enhancements.
- LCC web application technology and architecture planning discussions.
- Testing, regression validation, documentation, and report preparation.

Internship Work Report

Name: Navnit Kumar
College: VIT Bhopal
Project: OsdagWeb
Internship: FOSSEE Semester Long Internship (2026)

Date	Day	Task	Hours
05-Jan-2026	Monday	Explored OsdagWeb codebase and set up local development environment.	5
06-Jan-2026	Tuesday	Studied OsdagWeb repository structure and engineering module layout.	4
07-Jan-2026	Wednesday	Analysed moment connection module architecture across frontend and backend.	6
08-Jan-2026	Thursday	Reviewed Django adapter pattern used for OsdagWeb design modules.	5
09-Jan-2026	Friday	Fixed beam-to-beam end plate CAD rendering and connector section mapping.	4
10-Jan-2026	Saturday	Corrected beam-to-column end plate output and adapter integration issues.	6
11-Jan-2026	Sunday	Weekly Off.	0
12-Jan-2026	Monday	Resolved cover plate bolted connection module adapter defects.	4
13-Jan-2026	Tuesday	Fixed cover plate welded connection module parameter validation issues.	6
14-Jan-2026	Wednesday	Updated CAD model API section lists for end plate connection sessions.	5
15-Jan-2026	Thursday	Debugged end plate rendering logic in osdag_core CAD generation modules.	5
16-Jan-2026	Friday	Aligned end plate frontend output configs with backend result schemas.	4
17-Jan-2026	Saturday	Corrected bolted-to-end input configuration and validation rules.	6
18-Jan-2026	Sunday	Weekly Off.	0
19-Jan-2026	Monday	Updated welded-to-end output dock tables and result field mappings.	6
20-Jan-2026	Tuesday	Corrected simply supported beam adapter parameter submission logic.	4
21-Jan-2026	Wednesday	Built OnCantilever React workspace component and module configurations.	5
22-Jan-2026	Thursday	Consolidated module fixes and requested for review.	4
23-Jan-2026	Friday	Created purlin backend adapter and connected flexure_purlin design class.	6
24-Jan-2026	Saturday	Updated flexure and flexure_cantilever classes for web adapter compatibility.	5
25-Jan-2026	Sunday	Weekly Off.	0
26-Jan-2026	Monday	Fixed CadSceneContext state handling for multi-section CAD views.	5

27-Jan-2026	Tuesday	Resolved Output Dock image map inconsistencies across connection modules.	6
28-Jan-2026	Wednesday	Fixed design report CSV view paths for updated connection modules.	4
29-Jan-2026	Thursday	Reviewed LCC desktop workflow and mapped features to web components.	5
30-Jan-2026	Friday	Suggested authentication and project management approach for LCC web app.	4
31-Jan-2026	Saturday	Discussed strategy for LCC web platform with mentor.	6
01-Feb-2026	Sunday	Weekly Off.	0
02-Feb-2026	Monday	Prepared system context diagram suggestions for LCC web app modules.	6
03-Feb-2026	Tuesday	Reviewed LCC web app architecture notes and refined mentor suggestions.	4
04-Feb-2026	Wednesday	Compared LCC and OsdagWeb module patterns for reusable web architecture ideas.	5
05-Feb-2026	Thursday	Suggested component-based UI layout for LCC web application dashboards.	6
06-Feb-2026	Friday	Outlined API endpoint structure for LCC design and reporting services.	4
07-Feb-2026	Saturday	Summarized LCC web migration recommendations for mentor review.	5
08-Feb-2026	Sunday	Weekly Off.	0
09-Feb-2026	Monday	Built cached-model download utilities for BREP and STL export paths.	4
10-Feb-2026	Tuesday	Fixed BREP file reading compatibility with updated OpenCascade bindings.	5
11-Feb-2026	Wednesday	Implemented exportCad client method in engineering service hook layer.	4
12-Feb-2026	Thursday	Discussed technology stack options for LCC web application with mentor.	6
13-Feb-2026	Friday	Proposed React.js and Django-based architecture for LCC web app.	5
14-Feb-2026	Saturday	Corrected front, top, and side camera view mappings in CAD viewer.	6
15-Feb-2026	Sunday	Weekly Off.	0
16-Feb-2026	Monday	Suggested modular frontend structure for LCC web application screens.	5
17-Feb-2026	Tuesday	Recommended REST API design approach for LCC web backend services.	6
18-Feb-2026	Wednesday	Presented database and deployment considerations for LCC web platform.	4
19-Feb-2026	Thursday	Documented technology and architecture recommendations for LCC web app.	6
20-Feb-2026	Friday	Improved single-selection CAD part visibility in EngineeringModule panel.	4

21-Feb-2026	Saturday	Refactored UnifiedDropdownMenu default handler for graphics option routing.	5
22-Feb-2026	Sunday	Weekly Off.	0
23-Feb-2026	Monday	Participated in architecture review discussion for LCC web application.	4
24-Feb-2026	Tuesday	Debugged adapter exceptions reported during flexural module testing.	6
25-Feb-2026	Wednesday	Fixed Common.py utility functions affecting multiple design modules.	5
26-Feb-2026	Thursday	Updated column cover plate and column end plate design type modules.	4
27-Feb-2026	Friday	Corrected beam cover plate welded design class for web execution.	6
28-Feb-2026	Saturday	Improved SceneManager loading behaviour for connection CAD models.	4
01-Mar-2026	Sunday	Weekly Off.	0
02-Mar-2026	Monday	Updated DesignKeys and apiRoutes constants for flexural member modules.	5
03-Mar-2026	Tuesday	Prepared internship Task 1 chapter on moment connection module work.	4
04-Mar-2026	Wednesday	Drafted internship Task 2 chapter on tension member module fixes.	6
05-Mar-2026	Thursday	Prepared internship Task 3 chapter on flexural member development.	5
06-Mar-2026	Friday	Drafted internship Task 4 chapter on multi-format CAD export pipeline.	4
07-Mar-2026	Saturday	Prepared internship Task 5 chapter on navbar and UI improvements.	6
08-Mar-2026	Sunday	Weekly Off.	0
09-Mar-2026	Monday	Compiled conclusions and skills developed section for internship report.	5
10-Mar-2026	Tuesday	Exam Leave.	0
11-Mar-2026	Wednesday	Exam Leave.	0
12-Mar-2026	Thursday	Exam Leave.	0
13-Mar-2026	Friday	Exam Leave.	0
14-Mar-2026	Saturday	Exam Leave.	0
15-Mar-2026	Sunday	Weekly Off.	0
16-Mar-2026	Monday	Exam Leave.	0
17-Mar-2026	Tuesday	Exam Leave.	0
18-Mar-2026	Wednesday	Exam Leave.	0
19-Mar-2026	Thursday	Exam Leave.	0
20-Mar-2026	Friday	Updated keyboard shortcuts and background colour picker integration.	6
21-Mar-2026	Saturday	Performed testing across updated OsdagWeb design modules.	4
22-Mar-2026	Sunday	Weekly Off.	0
23-Mar-2026	Monday	Done moment and member module fixes.	5

24-Mar-2026	Tuesday	Performed regression testing across updated Osdag-Web design modules.	4
25-Mar-2026	Wednesday	Prepared internship technical documentation and report deliverables.	6
26-Mar-2026	Thursday	Validated design-to-CAD-to-output workflows for all modified modules.	5
27-Mar-2026	Friday	Re-tested moment connection modules after review feedback.	4
28-Mar-2026	Saturday	Verified cantilever and purlin modules through full design workflows.	6
29-Mar-2026	Sunday	Weekly Off.	0
30-Mar-2026	Monday	Tested UnifiedDropdownMenu Save 3D Model export submenu actions.	4
31-Mar-2026	Tuesday	Verified engineering workspace Quit action navigation to home page.	6
01-Apr-2026	Wednesday	Fixed tension member bolted module adapter and output configuration.	5
02-Apr-2026	Thursday	Resolved tension member welded module design workflow and CAD issues.	4
03-Apr-2026	Friday	Implemented simply supported beam module fixes in frontend and backend.	6
04-Apr-2026	Saturday	Developed cantilever flexural member submodule with adapter and service layer.	5
05-Apr-2026	Sunday	Weekly Off.	0
06-Apr-2026	Monday	Implemented purlin flexural member module and React workspace integration.	6
07-Apr-2026	Tuesday	Updated shared CAD scene manager, part config, and view mappings.	5
08-Apr-2026	Wednesday	Synchronized osdag_core design type modules with web adapter contracts.	4
09-Apr-2026	Thursday	Performed integration testing for moment connection design workflows.	6
10-Apr-2026	Friday	Validated tension member modules through design execution and CAD preview.	4
11-Apr-2026	Saturday	Tested flexural member modules and resolved Output Dock inconsistencies.	5
12-Apr-2026	Sunday	Weekly Off.	0
13-Apr-2026	Monday	Exam Leave.	0
14-Apr-2026	Tuesday	Exam Leave.	0
15-Apr-2026	Wednesday	Exam Leave.	0
16-Apr-2026	Thursday	Exam Leave.	0
17-Apr-2026	Friday	Exam Leave.	0
18-Apr-2026	Saturday	Exam Leave.	0
19-Apr-2026	Sunday	Weekly Off.	0
20-Apr-2026	Monday	Exam Leave.	0
21-Apr-2026	Tuesday	Exam Leave.	0
22-Apr-2026	Wednesday	Exam Leave.	0

23-Apr-2026	Thursday	Exam Leave.	0
24-Apr-2026	Friday	Exam Leave.	0
25-Apr-2026	Saturday	Exam Leave.	0
26-Apr-2026	Sunday	Weekly Off.	0
27-Apr-2026	Monday	Exam Leave.	0
28-Apr-2026	Tuesday	Addressed mentor review feedback on module adapter and CAD changes.	5
29-Apr-2026	Wednesday	Designed multi-format CAD export workflow for Save 3D Model menu.	4
30-Apr-2026	Thursday	Implemented backend exportCad API for BREP, STL, STEP, and IGES formats.	6
01-May-2026	Friday	Developed IFC export pipeline using IfcOpenShell mesh tessellation.	4
02-May-2026	Saturday	Integrated frontend export submenu options in UnifiedDropdownMenu.	6
03-May-2026	Sunday	Weekly Off.	0
04-May-2026	Monday	Tested cached and on-demand CAD export paths across design modules.	5
05-May-2026	Tuesday	Validated IFC export and mesh coarsening on complex connection models.	6
06-May-2026	Wednesday	Performed testing across updated OsdagWeb design modules.	4
07-May-2026	Thursday	Fixed navbar graphics options, camera views, and CAD part selection logic.	5
08-May-2026	Friday	Restored Create Design Report and OSI save actions in engineering workspace.	4
09-May-2026	Saturday	Prepared navbar and workspace UI improvements for mentor review.	6
10-May-2026	Sunday	Weekly Off.	0
11-May-2026	Monday	Performed testing across updated OsdagWeb design modules.	6
12-May-2026	Tuesday	Performed regression testing across updated OsdagWeb design modules.	4
13-May-2026	Wednesday	Reviewed code changes, mentor feedback, and pull request comments.	5
14-May-2026	Thursday	Prepared internship technical documentation and report deliverables.	4
15-May-2026	Friday	Finalized internship work report and consolidated project deliverables.	6

A.2 GitHub Contributions and Pull Requests

Throughout the internship, contributions were made to the OsdagWeb repository (`https://github.com/sogalabhi/Osdag-web`) through module development, bug fixes, CAD export pipeline implementation, and user interface improvements. These contributions were submitted and reviewed through multiple pull requests.

A.2.1 Major Pull Requests

- **#37 — Fix moment connection, tension member, simply supported, cantilever, and purlin modules**

  Consolidated fixes across 55 files (+2436 / -795) covering moment connection modules (beam-to-beam end plate, beam-to-column end plate, cover plate bolted/welded), tension member modules (bolted and welded), and flexural member modules (simply supported beam, cantilever, and purlin). Changes spanned frontend React configurations, Django backend adapters, shared CAD scene management, and `osdag_core` engineering modules. Merged on 25-April-2026.

- **#40 — Add Export BREP/STL/STEP/IGS/IFC options under navbar Save 3D Model**

  Implemented a multi-format CAD export pipeline with backend `exportCad` API support and frontend submenu integration under the "Save 3D Model" navbar action. Added on-demand model regeneration from input values, cached BREP/STL download utilities, and IFC export using IfcOpenShell mesh tessellation with adaptive mesh coarsening. Merged on 03-May-2026.

- **#87 — Fix navbar options**

  Refactored navbar menu handling for graphics options, camera view mappings (front/top/side), CAD part selection logic, Create Design Report action, and OSI save workflow. Updated keyboard shortcuts and background colour picker integration in the shared engineering workspace. Submitted for mentor review; closed pending product alignment with the revamped CAD system.

A.2.2 LCC Web Application Architecture Contributions

In addition to OsdagWeb development, contributions were made toward planning the LCC web application through discussions and architectural suggestions shared with the mentor. This work included:

- Technology stack recommendations for the LCC web platform.
- Proposed React.js and Django-based full-stack architecture.
- Suggested modular frontend structure and REST API design for backend services.
- Database, deployment, and project management considerations.
- CAD visualization strategy and component-based UI layout suggestions.
- Mapping of LCC desktop workflows to proposed web application modules.

A.2.3 Summary of Contributions

The internship contributions resulted in substantial improvements to the OsdagWeb platform, including structural design module stabilization, flexural member module integration, multi-format CAD export capability, and engineering workspace UI refinements.

The completed work established a foundation for:

- Reliable browser-based steel connection and member design workflows.
- Consistent frontend-backend module integration through adapter patterns.
- On-demand and cached CAD model export in industry-standard formats.
- Improved navbar and CAD viewer usability in the shared engineering workspace.
- BIM interoperability through IFC export support.
- Continued web-platform development for OsdagWeb and architectural planning for LCC.

These contributions enhanced the functionality, usability, maintainability, and interoperability of the OsdagWeb project under the FOSSEE Semester Long Internship (2026).

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.