FOSSEE Summer Internship Report

On

Design and Development of Website version for Osdag

Submitted by

Pramila Kumari

3rd Year B.Tech Student, Department of Electronics and Communication

Banasthali Vidyapith University

Rajasthan

Under the Guidance of

Prof. Siddhartha Ghosh

Department of Civil Engineering

Indian Institute of Technology Bombay

Mentors:

Ajmal Babu M S

Parth Karia

Ajinkya Dahale

August 8, 2025

Acknowledgments

I would like to express my sincere gratitude to everyone who supported me during my internship with the Osdag team under the FOSSEE Summer Fellowship 2025.

I am deeply thankful to the Osdag team members — Mr. Ajmal Babu M. S., Mr. Ajinkya Dahale, and Mr. Parth Karia — for their continuous guidance, valuable feedback, and encouragement throughout the project. I also extend my appreciation to Prof. Siddhartha Ghosh, Principal Investigator of Osdag, Department of Civil Engineering, IIT Bombay, for providing the opportunity to contribute to this initiative.

I am grateful to Prof. Kannan M. Moudgalya, Principal Investigator of the FOSSEE Project, Department of Chemical Engineering, IIT Bombay, for his vision and leadership, as well as to FOSSEE Managers Ms. Usha Viswanathan and Ms. Vineeta Parmar, and their entire team, for their constant support.

I would like to acknowledge the support from the National Mission on Education through Information and Communication Technology (ICT), Ministry of Education (MoE), Government of India, for facilitating this project.

Finally, I thank my fellow interns and colleagues for their collaboration, as well as my college and department for their encouragement during my studies.

Contents

1 Introduction **4**

1.1 National Mission in Education through ICT 4

1.1.1 ICT Initiatives of MoE 5

1.2 FOSSEE Project 6

1.2.1 Projects and Activities 6

1.2.2 Fellowships 6

1.3 Osdag Software 7

1.3.1 Osdag GUI 8

1.3.2 Features 8

2 Screening Task **9**

2.1 Problem Statement 9

2.2 Tasks Done 9

3 Internship Task 1: Simply Supported Beam Module Integration in Osdag Web Application **12**

3.1 Task 1: Problem Statement 12

3.2 Task 1: Tasks Done 12

3.3 Task 1: Code Overview 13

3.3.1 Frontend (React JS) 13

3.3.2 Backend (Django) 14

3.3.3 Logic Explanation 15

3.4 Task 1: Documentation 15

3.4.1 Directory Structure 15

3.4.2 Deprecation of Legacy Design Structure 16

4 Internship Task 2: Restructuring the Simply Supported Beam Module (Frontend) **17**

4.1 Task 2: Problem Statement 17

4.2 Task 2: Tasks Done 17

4.3 Task 2: Code and Structure 18

4.3.1 Frontend (React JS) 18
4.3.2 Directory Structure (After Refactor) 19
4.3.3 Logic Explanation 19
4.4 Task 2: Summary of My Contribution 19

5 Conclusions **20**
5.1 Tasks Accomplished 20
5.2 Skills Developed 21

A Appendix **22**
A.1 Work Reports 22

Bibliography **25**

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.

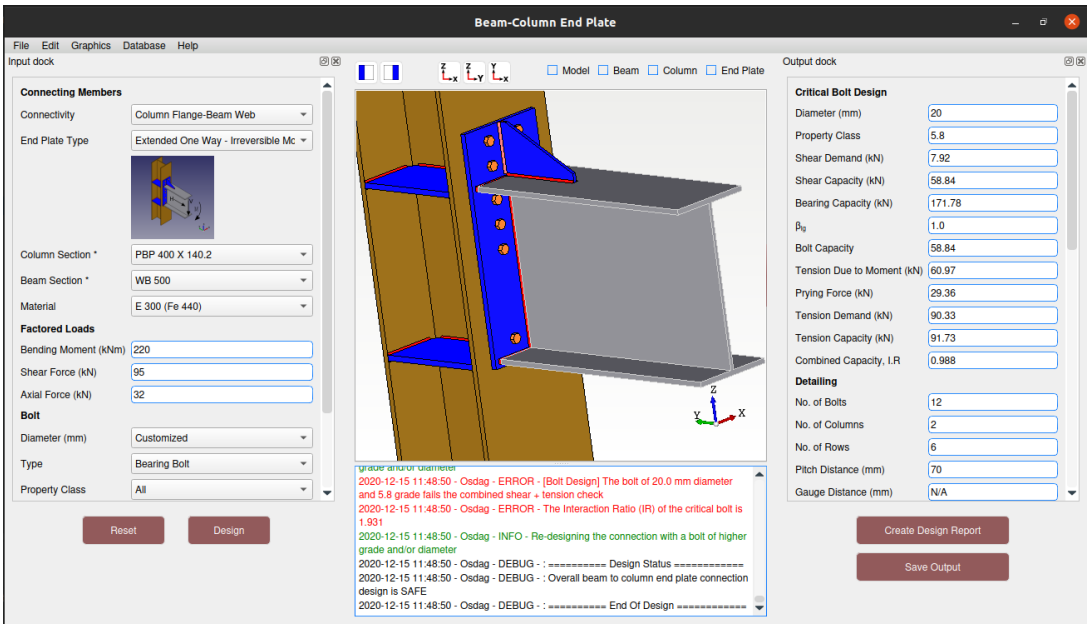

Figure 1.2: Osdag GUI

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 with contributing to the web app development of Osdag. The objective was to implement a structural steel design module from the desktop version into the cloud-based Osdag Web Application. The selected module for this task was the **Simply Supported Beam** under the Flexural Member category. The goal was to replicate the IS 800:2007 design logic from Osdag Desktop, develop a fully functional frontend using React, implement RESTful APIs with Django for backend calculations, and ensure smooth frontend-backend integration with PDF report generation.

2.2 Tasks Done

- **Setup and Deployment:**
  - Installed and configured Ubuntu development environment.
  - Cloned Osdag Web repository and all related submodules.
  - Resolved multiple dependency issues for both frontend (Node.js/NPM) and backend (Django/FreeCAD).
  - Verified that both servers ran locally without errors and supported full-stack development.
- **Module Selected:** Implemented the **Flexural Member – Simply Supported Beam** module, adapting the desktop logic for the web platform. The module

performs beam design checks according to IS 800:2007 and generates detailed calculation reports.

- **Frontend Development:**
  - Built using React with a modular component structure (`InputForm.jsx`, `OutputDock.jsx`, `Logs.jsx`, `ReportModal.jsx`).
  - Designed a clean and responsive UI with dropdowns, number fields, and selection menus for beam parameters.
  - Integrated validation to prevent incorrect or incomplete inputs.
  - Connected the frontend to backend APIs using centralized functions in `ModuleContext`.
  - Results are displayed dynamically in the Output Dock after successful calculation.
- **Backend Development:**
  - Developed Django REST API endpoints for design calculation and report generation.
  - Implemented Python logic for Simply Supported Beam design (moment capacity, shear checks, buckling checks) as per IS 800:2007.
  - Integrated LaTeX-based PDF report generator to compile and return calculation results.
  - Added session state handling to maintain consistent data flow between frontend and backend.
- **Integration:**
  - The frontend submits user input to the backend via the `createDesign` API call.
  - Backend processes the calculation and returns structured JSON results.
  - The frontend updates the Output Dock and allows users to download the generated PDF report.
- **Code Structure:** Maintained a modular and organized directory layout:

```
 osdagclient/
src/
    modules/
        flexuralMember/
            simplySupportedBeam/
                components/
                   SimplySupportedBeamOutputDock.jsx
                configs/
                   simplySupportedBeamconfig.js
                   simplySupportedBeamOutputconfig.js
                index.js
                SimplySupportedBeam.jsx
```

- **Documentation:**
  - **Methodology:** Started with setting up the Osdag Web environment, studied the desktop Simply Supported Beam logic, planned the UI/UX, implemented backend logic and API endpoints, and integrated frontend-backend workflow.
  - **Challenges:** Handling dependency issues, aligning calculation outputs with UI, ensuring state persistence across inputs, and debugging API integration.
  - **References:** Osdag desktop source code, IS 800:2007 standard, Django REST Framework documentation, React.js documentation.

Chapter 3

Internship Task 1: Simply Supported Beam Module Integration in Osdag Web Application

3.1 Task 1: Problem Statement

The goal of this task was to implement a new design module for **Simply Supported Beam** under the Flexural Member category in the Osdag Web Application. The module was to replicate the design logic from the Osdag Desktop version (based on IS 800:2007) in a full-stack web application using React for the frontend and Django for the backend. The web solution should allow users to:

- Enter beam geometry, section properties, and loading parameters via a browser-based form.
- Trigger backend computations to perform design checks according to IS 800:2007.
- Display results in a structured output dock and generate a downloadable design report.

3.2 Task 1: Tasks Done

This work involved:

- Setting up the Osdag Web environment (frontend and backend) and resolving dependency issues.
- Reviewing the Simply Supported Beam logic from Osdag Desktop and mapping it to a web workflow.
- Creating Django REST API endpoints for the design logic.
- Developing a responsive React interface for data entry and results viewing.
- Managing API requests/responses using React state management and a centralized ModuleContext.
- Implementing validation and error handling for user inputs.
- Integrating the LaTeX-based PDF report generator to produce downloadable reports after computation.
- Conducting full end-to-end testing and fixing bugs discovered during the review phase.

3.3 Task 1: Code Overview

3.3.1 Frontend (React JS)

The React frontend collects form data and communicates with the Django backend.

Listing 3.1: Triggering Simply Supported Beam Design API Call

```
import { useModuleContext } from '../context/ModuleContext';

const { createDesign } = useModuleContext();

const handleSubmit = async () => {
  const payload = {
    module_id: "flexural_member_simply_supported_beam",
    inputs: {
      section: "ISMB 300",
      span_length: 6000,
```

```
      load_type: "UDL",
      load_value: 20
    }
  };
  const result = await createDesign(payload);
  // Update UI with the results
};
```

Key Features

- Centralized API call using ModuleContext for consistency.
- Inputs structured with module ID and parameters for backend processing.
- State updates trigger re-rendering of OutputDock with computed results.

3.3.2 Backend (Django)

The Django backend acts as the computation engine, parsing input, performing IS 800:2007 checks, and returning design data.

Listing 3.2: Django API View for Simply Supported Beam

```
from rest_framework.decorators import api_view
from rest_framework.response import Response
from .beam_design import run_beam_design

@api_view(['POST'])
def calculate_output(request, module_id):
    inputs = request.data['inputs']
    section = inputs['section']
    span = inputs['span_length']
    load_type = inputs['load_type']
    load_value = inputs['load_value']

    result = run_beam_design(section, span, load_type, load_value
        )
```

```
    return Response(result)
```

Other API Endpoints

- `/generate-report/` — compiles a PDF report with design results.

3.3.3 Logic Explanation

- API views are handled using Django REST Framework.
- Input data is validated and passed to the Python calculation logic.
- IS 800:2007 procedures are implemented to determine moment capacity, shear capacity, and deflection.
- After computation, results are sent back as JSON to the frontend.

3.4 Task 1: Documentation

3.4.1 Directory Structure

```
osdagclient/
  src/
      modules/
          flexuralMember/
              simplySupportedBeam/
                  components/
                     SimplySupportedBeamOutputDock.jsx
                  configs/
                     simplySupportedBeamconfig.js
                     simplySupportedBeamOutputconfig.js
                  index.js
                  SimplySupportedBeam.jsx
```

3.4.2 Deprecation of Legacy Design Structure

The Simply Supported Beam design logic previously existed only in Osdag Desktop as a monolithic Python script. The new web implementation modularizes both frontend and backend, enabling easier maintenance, scalability, and remote access via browsers.

Chapter 4

Internship Task 2: Restructuring the Simply Supported Beam Module (Frontend)

4.1 Task 2: Problem Statement

After the initial implementation of the **Simply Supported Beam** module in Osdag Web, the frontend codebase had become monolithic and difficult to maintain. The goal of this task was to refactor the frontend into a clean, modular, and reusable structure while preserving existing functionality. This restructuring aimed to:

- Improve readability and maintainability of the code.
- Separate UI concerns into dedicated components.
- Centralize API calls for reusability and easier debugging.
- Ensure the UI remained responsive and visually consistent after refactoring.

4.2 Task 2: Tasks Done

This work involved:

- Creating a new, organized directory structure under `osdagclient/src/modules/flexural_member/simply_supported_beam/`.

- Moving all input handling logic into a dedicated `InputForm.jsx` component.
- Extracting results display into a dedicated `OutputDock.jsx` component.
- Migrating the report generation modal into a `ReportModal.jsx` component.
- Centralizing API calls (`createSession`, `createDesign`) in `ModuleContext`.
- Removing duplicate logic and simplifying state management.
- Testing the UI for responsiveness and bug-free integration with backend APIs.

4.3 Task 2: Code and Structure

4.3.1 Frontend (React JS)

The refactoring process modularized the Simply Supported Beam frontend into smaller components.

Listing 4.1: Refactored API Call in ModuleContext

```
export const createDesign = async (payload) => {
  try {
    const response = await fetch('/api/design/', {
      method: 'POST',
      headers: {'Content-Type': 'application/json'},
      body: JSON.stringify(payload),
    });
    return await response.json();
  } catch (error) {
    console.error("Design API call failed", error);
    return null;
  }
};
```

Key Features

- All API calls are now handled from a single context file.

- Components can import API functions without duplicating request logic.
- The design results are rendered directly by `OutputDock.jsx`.

4.3.2 Directory Structure (After Refactor)

```
osdagclient/
  src/
      modules/
          flexuralMember/
              simplySupportedBeam/
                  components/
                     SimplySupportedBeamOutputDock.jsx
                  configs/
                     simplySupportedBeamconfig.js
                     simplySupportedBeamOutputconfig.js
                  index.js
                  SimplySupportedBeam.jsx
```

4.3.3 Logic Explanation

- The `InputForm.jsx` handles all parameter inputs and validation.
- The `OutputDock.jsx` dynamically renders the backend results.
- The `ReportModal.jsx` manages LaTeX report generation and download.
- All API requests go through centralized functions for easy debugging and changes.

4.4 Task 2: Summary of My Contribution

In this task, I reorganized the frontend of the Simply Supported Beam module into modular, reusable components. This refactor improved maintainability, reduced redundancy, and ensured consistent integration with the backend API, making future updates easier and less error-prone.

Chapter 5

Conclusions

5.1 Tasks Accomplished

During my internship, I worked on developing the **Simply Supported Beam** module for Osdag-Web. Some of the main things I accomplished include:

- **Setting up the development environment** — Installed Ubuntu, configured VS Code and Git, set up Osdag desktop and Osdag-Web, and fixed installation issues for both frontend and backend.
- **Exploring the project** — Studied the Osdag codebase, project structure, and existing modules to understand how the frontend and backend interact.
- **Frontend development for the module** — Built the input UI for the Simply Supported Beam module in React, including dropdowns and fields for design parameters. Created an output dock to display results and later refactored the UI into smaller components like `OutputDock.jsx` and `ReportModal.jsx`.
- **Backend calculation logic** — Implemented the IS 800:2007-based calculation logic in Python and created APIs for the module. Added session handling and input validation to ensure correct data flow.
- **Refactoring for better structure** — Organized the code into a clean directory layout, moved logic into separate JSX files, and updated API calls to use a centralized context.

- **Collaboration and feedback** — Worked with teammates to improve the module's UI responsiveness and made changes suggested by mentors.

5.2 Skills Developed

Through this internship, I improved in many technical and collaborative areas:

- **React.js frontend development** — Learned to build reusable components and connect them to backend APIs.
- **Backend with Python/Django** — Understood how to structure logic, build APIs, and integrate them with frontend requests.
- **Following design codes** — Applied IS 800:2007 standards in structural design calculations.
- **Code organization and refactoring** — Learned to make code more maintainable and modular.
- **Problem-solving** — Improved my ability to resolve setup, integration, and logical issues.
- **Collaboration** — Gained experience working in a team and implementing mentor feedback.

Overall, this internship taught me how to work on a real-world full-stack web application module from scratch. I learned how to connect different parts of a project — frontend, backend, and calculations — and deliver a complete, functional feature.

Chapter A

Appendix

A.1 Work Reports

Internship Work Report

Name: Pramila Kumari
Project: Osdag
Internship: FOSSEE Summer Fellowship 2025

Date	Day	Task	Hours Worked
16-May-2025	Friday	Fellowship orientation. Installed Ubuntu and set up VS Code and Git.	6
17-May-2025	Saturday	Installed Osdag and required dependencies.	5
18-May-2025	Sunday	Debugged errors during Osdag installation.	5
19-May-2025	Monday	Cloned Osdag-web repo and submodules. Began installing dependencies.	4
20-May-2025	Tuesday	Resolved dependency issues and successfully ran the frontend.	4
21-May-2025	Wednesday	Reviewed project documentation and explored Osdag features.	5
22-May-2025	Thursday	Successfully ran backend after resolving issues.	5
23-May-2025	Friday	Analyzed architectural flow and directory structure of Osdag.	5
24-May-2025	Saturday	Understood the codebase of existing modules.	5
25-May-2025	Sunday	Began working on Simply Supported Beam module in Flexural Member.	4
26-May-2025	Monday	Understood existing code and desktop UI/UX for frontend module.	4
27-May-2025	Tuesday	Continued development of Simply Supported Beam module.	4
28-May-2025	Wednesday	Debugged frontend issues.	5
29-May-2025	Thursday	Built UI input section including dropdowns and design parameter fields.	5
30-May-2025	Friday	Completed frontend output dock for the module.	4
31-May-2025	Saturday	Started backend calculation logic development.	5
01-Jun-2025	Sunday	Implemented core design logic for strut based on IS 800:2007.	4
02-Jun-2025	Monday	Explored backend Python code and learned React.	5
03-Jun-2025	Tuesday	Developed backend API endpoint.	5
04-Jun-2025	Wednesday	Implemented session state and input validation logic.	4
05-Jun-2025	Thursday	Developed API to interface with FreeCAD.	5
06-Jun-2025	Friday	Generated 3D CAD model for strut design.	6
07-Jun-2025	Saturday	Implemented backend logic in reportGenerator_latex.py.	5
08-Jun-2025	Sunday	Reviewed React component composition.	6
09-Jun-2025	Monday	Created OutputDock and Logs components.	5
10-Jun-2025	Tuesday	Connected frontend to backend API for calculation.	5
11-Jun-2025	Wednesday	Integrated CAD viewer to display 3D model.	6
12-Jun-2025	Thursday	Integrated backend report generation API.	5
13-Jun-2025	Friday	Performed full module testing and bug fixing.	5

14-Jun-2025	Saturday	Debugged and resolved backend issues.	6
15-Jun-2025	Sunday	Weekend self-study.	6
16-Jun-2025	Monday	Documented module and reviewed with mentor.	6
17-Jun-2025	Tuesday	Improved frontend as per mentor's feedback.	6
18-Jun-2025	Wednesday	Continued backend debugging.	6
19-Jun-2025	Thursday	Documented and held second review with mentor.	5
20-Jun-2025	Friday	Worked on backend issues.	5
21-Jun-2025	Saturday	Debugged backend errors.	6
22-Jun-2025	Sunday	Fully integrated backend.	5
23-Jun-2025	Monday	Resolved calculation logic issues.	5
24-Jun-2025	Tuesday	Created structured directory under osdagclient/src/modules/flexural_member/simply_supported_beam.	6
25-Jun-2025	Wednesday	Refactored input logic into separate JSX files.	6
26-Jun-2025	Thursday	Extracted result logic into OutputDock.jsx.	7
27-Jun-2025	Friday	Migrated report modal to ReportModal.jsx.	4
28-Jun-2025	Saturday	Refactored API calls using ModuleContext.	4
29-Jun-2025	Sunday	Validated full refactored workflow.	5
30-Jun-2025	Monday	Debugged final UI issues and completed with team help.	5

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.