FOSSEE Winter Internship Report

On

Development of Algorithm for Lacing Module and Documentation on Report Generation for Osdag

Submitted by

Shubham Kumar

3rd Year B.Tech Student, Department of Civil and Infrastructure Engineering

Indian Institute of Technology Jodhpur

Jodhpur 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

January 5, 2025

Acknowledgments

- I would like to sincerely thank the **Osdag and FOSSEE teams** at **IIT Bombay** for giving me this wonderful opportunity. Through this internship, I was able to learn new things, and it was a truly great experience for me.

- A special thanks to our mentors, **Parth Karia, Ajinkya Dahale, and Ajmal Babu M. S.**, project staff at the Osdag team, for their constant support during the internship and for teaching us new skills that have greatly enriched our learning experience.

- I am deeply grateful to **Prof. Siddhartha Ghosh,** Principal Investigator (PI) of the Osdag Project and Professor in the Department of Civil Engineering at IIT Bombay, for his invaluable guidance and support throughout the project.

- I extend my sincere thanks to **Prof. Kannan M. Moudgalya**, Principal Investigator (PI) of the FOSSEE Project, Department of Chemical Engineering, IIT Bombay, for providing me with this opportunity and for his continuous encouragement.

- My special thanks to FOSSEE Managers, **Ms. Usha Viswanathan and Ms. Vineeta Parmar,** and their entire team for their support and coordination, which greatly facilitated the project's progress.

- I extend my heartfelt thanks to my colleagues **Amrutha, Anuanjani,** and **Debayan** for their help and collaboration throughout this internship.

- I also want to express my gratitude to my **professors** from the **Civil and Infrastructure Engineering Department at IIT Jodhpur** for their guidance and support during my academic journey.

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.

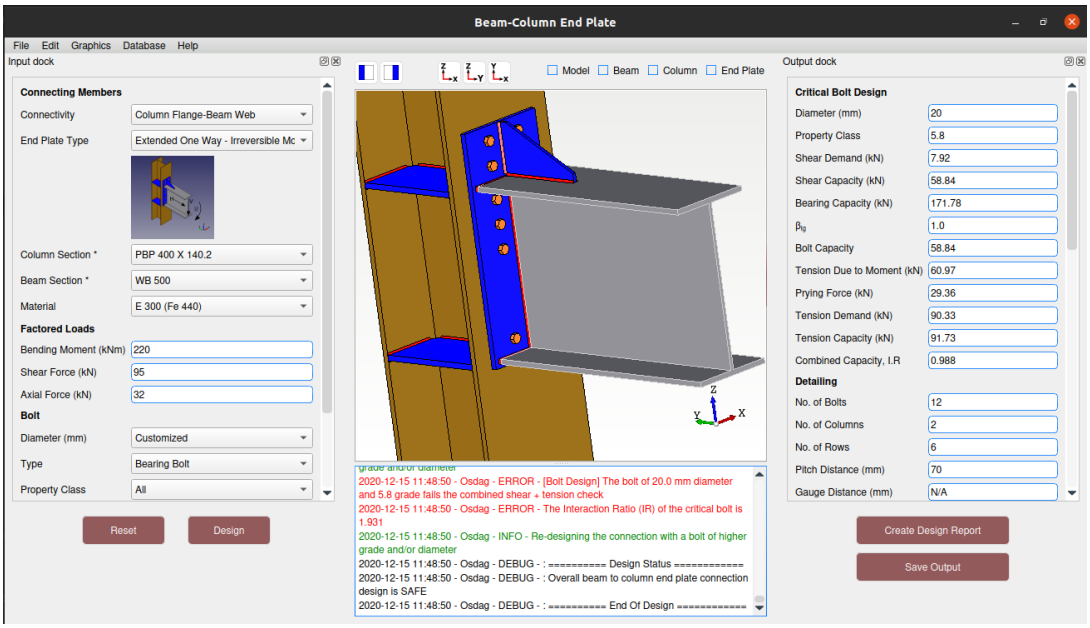

Figure 1.2: Osdag GUI

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

The task involves preparing a Python program to facilitate the **design of flexural members**, specifically those that are laterally supported and subjected to low shear conditions. The program should include the following key functionalities:

Design Checks: Perform comprehensive design checks based on the guidelines specified in IS 800:2007. These checks should ensure the structural integrity and safety of the flexural member under given loading conditions.

Input Handling: Accept input parameters such as the shear force, bending moment, and effective span of the member from a designated input text file.

Output Generation: Produce detailed design results and assessments in an organized output text file.

Documentation: Generate a complete report summarizing the design procedure, input parameters, output results, and adherence to IS 800:2007 standards.

The deliverables for this task include:

-The Python program developed for the design process.

-Input and output text files demonstrating the program's functionality.

-A comprehensive report detailing the design methodology and results.

2.2 Tasks Done

Comprehensive Review of IS:800:2007 The task began with a thorough study of the Indian Standard IS:800:2007, which provides guidelines for the design of steel structures. This knowledge formed the foundation for developing the program, ensuring compliance with the safety and design criteria specified in the standard.

Development of a Python Program A Python program was developed to assess the structural capacity of beams based on the given input parameters. The key functionalities of the program included:

Input Reading: The program reads beam design parameters from a formatted text file, ensuring flexibility and reusability.

Inputs include span length, moment, shear force, yield strength, section modulus, moment of inertia, and cross-sectional area.

Capacity Analysis: Functions were created to check the moment capacity, shear capacity, and deflection limits of the beam.

The calculations adhered to the IS:800:2007 guidelines, incorporating safety factors and material constants.

Output Generation: After performing the analysis, the program outputs the results to a text file, detailing whether the beam satisfies the moment, shear, and deflection criteria for each test case.

Validation of Input and Output Formats

Ensured that the input file format was correctly structured, including the mandatory fields and their respective units.

Any mismatch in column names or missing data was accounted for to avoid processing errors.

The output file was designed to be user-friendly, summarizing the applied values, calculated capacities, and status (Pass/Fail) for each parameter.

Incorporation of Design Constants and Factors

The program included pre-defined constants such as the modulus of elasticity (200 GPa for steel) and a partial safety factor for material strength ($\text{gamma}_m o = 1.1$).

$These values could be modified to meet specific design requirements.$

Execution and Debugging

The program was executed for multiple test cases, ensuring accurate results across different scenarios. Any identified errors or inconsistencies were resolved to refine the program.

Preparation for Beam Analysis Tool Integration

As a personal initiative (not required as part of the screening task), additional efforts were made to extend the program's functionality. This included laying the groundwork for generating default input files and preparing for potential integration with other beam analysis tools.

2.3 Python Code for Beam Design Checks

This section presents the Python code written for the screening task, along with a detailed explanation of its functionality. The code is designed to perform moment, shear, and deflection checks for beams based on the IS:800:2007 standards.

2.4 Python Code for Beam Design Checks

This section presents the Python code written for the screening task, along with a detailed explanation of its functionality. The code is designed to perform moment, shear, and deflection checks for beams based on the IS:800:2007 standards.

2.4.1 Python Code

```
# Design Check For Beams

import math

# Constants and design factors from IS:800:2007
E = 200000  # Modulus of elasticity in MPa (for steel)
gamma_mo = 1.1  # Partial safety factor for material strength

def read_input(file_path):
    with open(file_path, 'r') as f:
        data = f.readlines()
```

```
    # Process data to handle multiple test cases
    inputs_list = []
    inputs = {}

    for line in data:
        if "Test Case" in line:
            if inputs:
                inputs_list.append(inputs)
                inputs = {}
        elif line.strip():
            key, value = line.strip().split(':')
            value = ''.join(filter(lambda x: x.isdigit() or x ==
               '.', value.strip()))
            inputs[key.strip()] = float(value)

    if inputs:
        inputs_list.append(inputs)

    return inputs_list

def moment_capacity_check(Z, fy, beta_b=1.0, gamma_mo=1.1):
    Md = beta_b * Z * fy / gamma_mo
    return Md

def shear_capacity_check(shear_force, area, fy):
    Vd = 0.6 * area * fy / gamma_mo
    return Vd, Vd >= shear_force

def deflection_check(span, moment, I):
    delta = (5 * moment * span**2) / (48 * E * I)
    max_deflection = span / 250
    return delta, delta <= max_deflection
```

```
def design_flexural_member(inputs):
    span = inputs["Span"]
    moment = inputs["Moment"]
    shear_force = inputs["Shear Force"]
    fy = inputs["Yield Strength"]
    Z = inputs["Section Modulus"]
    I = inputs["Moment of Inertia"]
    area = inputs["Cross-sectional Area"]

    results = {}

    Md = moment_capacity_check(Z, fy)
    moment_limit = 1.2 * Z * fy / gamma_mo
    failure_status = Md < moment and Md < moment_limit

    if failure_status:
        if Md < moment:
            reason = "Moment capacity is less than the applied
               moment."
        if Md < moment_limit:
            reason = "Moment capacity is within the permissible
               limit."
    else:
        reason = "Moment capacity is sufficient to resist the
           applied moment."

    results['Moment Capacity Check'] = {
        'Applied Moment (kN m)': moment,
        'Moment Capacity (kN m)': Md,
        'Status': 'Fail' if failure_status else 'Pass',
        'Reason': reason
    }
```

```
    Vd, shear_status = shear_capacity_check(shear_force, area, fy
        )
    results['Shear Capacity Check'] = {
        'Applied Shear Force (kN)': shear_force,
        'Shear Capacity (kN)': Vd,
        'Status': 'Pass' if shear_status else 'Fail'
    }

    delta, deflection_status = deflection_check(span, moment, I)
    results['Deflection Check'] = {
        'Calculated Deflection (mm)': delta,
        'Permissible Deflection (mm)': span / 250,
        'Status': 'Pass' if deflection_status else 'Fail'
    }

    return results

def write_output(results_list, file_path):
    with open(file_path, 'w') as f:
        for i, results in enumerate(results_list, 1):
            f.write(f"Results for Test Case {i}:\n")
            for check, details in results.items():
                f.write(f"{check}:\n")
                for key, value in details.items():
                    f.write(f"  {key}: {value}\n")
                f.write("\n")
            f.write("\n")

if __name__ == "__main__":
    try:
        inputs_list = read_input('/content/input.txt')
        results_list = [design_flexural_member(inputs) for inputs
            in inputs_list]
        write_output(results_list, 'beam_design_output.txt')
```

```
        print("Output file 'beam_design_output.txt' has been
            created successfully.")
    except Exception as e:
        print(f"An error occurred: {e}")
from google.colab import files
files.download('beam_design_output.txt')
```

[**Python Code for Beam Design Checks**]

2.4.2 Explanation of the Code

The code is structured as follows:

- **Constants and Factors:** Key constants such as the modulus of elasticity for steel (200 GPa) and the partial safety factor ($\gamma_{mo} = 1.1$) are defined.
- **Input Reading:** The `read_input` function parses a text file containing beam parameters for multiple test cases and organizes them into dictionaries.
- **Capacity Checks:** Three functions (`moment_capacity_check`, `shear_capacity_check`, and `deflection_check`) calculate the moment, shear, and deflection capacities, comparing them with applied values to determine the beam's safety.
- **Design Analysis:** The `design_flexural_member` function aggregates the checks and prepares a results dictionary for each test case.
- **Output Writing:** The `write_output` function generates a summary of results for all test cases in a formatted text file.
- **Main Execution Block:** Reads input, executes the analysis, and writes output while handling potential errors.

graphicx

Chapter 3

Internship Task 1 : Development of Lacing Module

3.1 Task 1: Problem Statement

The problem involved creating a structured flowchart to develop an algorithm for the lacing module in Osdag. The task required defining a logical sequence of steps: determining input parameters, evaluating conditions, and establishing decision points.

3.2 Task 1: Tasks Done

To develop the algorithm for the lacing module in Osdag, the following methodologies and processes were implemented: **1. Research and Standards Compliance** The development process was guided by the provisions of IS 800:2007 and the Osdag DDCL (Design Details Checklist Logic). These resources provided the technical framework for checks, calculations, and design requirements essential for the algorithm.

2. Flowchart Development A detailed flowchart was created using Canva to define the algorithm's logical sequence. The flowchart (as attached) includes the following elements: Inputs and Initialization: User provides inputs such as material grade, section size, and end conditions.

Section Classification: Determines if the section is compact, semi-compact, or plastic, following specific conditions.

Design Checks and Calculations: Includes checks for slenderness ratio, angle, spacing,

and compressive strength calculations.

Iterative Decision Points: Conditions like whether the section meets design requirements or needs to be revised, ensuring all constraints are satisfied.

Output: Generates comprehensive results, including dimensions, forces, and connection details for lacing.

3. Implementation Strategy The algorithm incorporated critical calculations such as:

Effective Area: Based on the steel table and effective area parameter.

Compressive and Tensile Strength Checks: Satisfied using equations from the DDCL.

Connection Design: Covered both bolted and welded options, ensuring compliance with standards.

For each iteration, feedback loops were established to refine the output based on design conditions.

4. Validation and Documentation

The flowchart was iteratively reviewed to address potential issues and ensure completeness. Documentation for each step was detailed for future reference and user understanding.

3.3 Task 1: Flowchart

Description of the Flowchart

The flowchart for the **Lacing Module Algorithm in Osdag** represents a comprehensive sequence of steps and decision points to ensure the design meets structural and code requirements. Below is a detailed explanation of its components:

1. Input Section

- **Materials and Section Data**: Inputs include material grade, section size, and unsupported lengths in y and z directions.
- **Lacing Profile and Pattern**: Specifies the lacing type (flat, angle, or channel) and pattern (single or double).

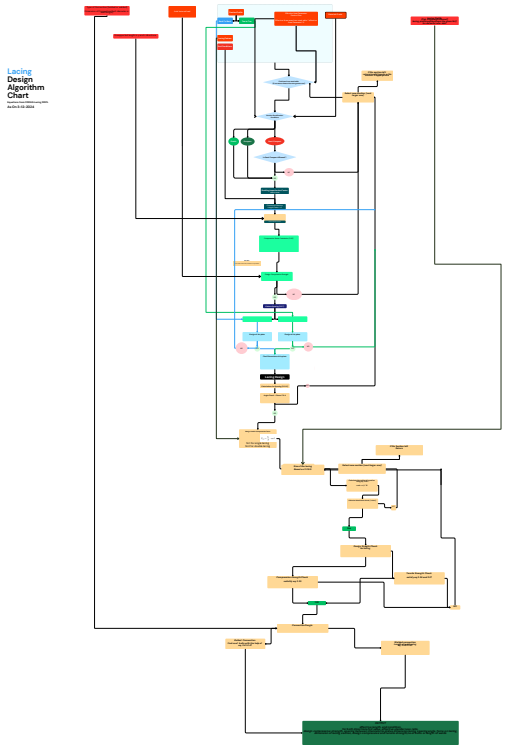

Figure 3.1: Flowchart for Lacing Module Algorithm

- **End Conditions and Connection Type**: Users define the type of connection (bolted or welded) and the effective length factor (K).

2. Section Classification

- Evaluates whether the section is plastic, semi-compact, or compact based on IS 800:2007.
- If the section is unsuitable(either slender or semi-compact(if its not allowed)), the algorithm selects the next larger section or returns an error if no section is left.

3. Design Calculations

- **Effective Area and Compressive Stress**: Calculations are performed to determine the effective area and compressive stress using equations from the DDCL.
- **Slenderness Ratio Check**: Ensures compliance with slenderness limits.
- **Spacing and Angle Checks**: Verifies spacing between lacings and checks the angle against standard requirements.

4. Lacing and Connection Design

- **Lacing Design**: Determines dimensions and spacing of lacing elements based on load and slenderness requirements.
- **Connection Design**: For bolted connections, calculates the number of bolts needed using relevant equations. For welded connections, computes the required weld length.

5. Iterative Decision Points

- Includes multiple checks such as: Does the design meet compressive and tensile strength requirements? Are the spacing and angle within allowable limits?
- If a check fails, the algorithm adjusts parameters or selects a new section.

6. Output Section

Provides detailed results, including:

- Effective length and end conditions.
- Design compressive strength and effective slenderness ratio.
- Dimensions of tie plates and lacings, angle specifications, and connection details (bolts or welds).

Chapter 4

Internship Task 2: Report Generation in OSDAG

4.1 Task 2: Problem Statement

Develop and debug the report generation program for the compression module and column module, and document the process of report generation in OSdag(How report is being generated in Osdag and what commands/files are referred).

4.2 Task 2: Tasks Done

Debugged the program using DDCL and IS 800:2007 standards, and developed the corrected program for report generation in the column and compression modules. Studied the process in other modules and applied similar methods to these modules. Additionally, created documentation detailing the report generation process.

4.3 Task 2: Documentation

Attached is the PDF file detailing the report generation process. Click here to view the report generation process.

Chapter 5

Internship Task 3: Development of Purlin Module

5.1 Task 3: Problem Statement

The task involved assisting in programming the Python code for the Purlin module in Osdag.

5.2 Task 3: Tasks Done

Studied the Purlin module using DDCL and contributed to developing functions for design checks, including moment check, shear check, and web resistance checks. Also worked on section classification and compiled details necessary for output documentation and report generation.

Chapter 6

Conclusions

6.1 Tasks Accomplished

The algorithm for the lacing module in Osdag has been successfully completed. Fixes were implemented in the report generation feature for both column and compression modules. Additionally, comprehensive documentation on the report generation process in Osdag was created. Progress was also made on the purlin module program, which is now nearly complete.

6.2 Skills Developed

During the fellowship, I acquired basic knowledge of GitHub for version control and collaboration and developed skills in using LaTeX for creating professional documentation. My Python programming skills were significantly enhanced, and I improved my design capabilities using Canva. Additionally, I honed teamwork and coordination skills, which have been essential for collaborative projects.

Chapter A

Appendix

A.1 Work Reports

Internship Work Report

Name: Shubham Kumar
Project: Osdag
Internship: FOSSEE Winter Fellowship 2024

Work Log

Date	**Day**	**Task**	**Hours Worked**
13-Nov-2024	Wednesday	Studied the basics of the steel engineering and IS Code 800 2007	4
14-Nov-2024	Thursday	Tried installing the osdag in the windows 10	4
15-Nov-2024	Friday	Meeting regarding the development of lacing module	3
18-Nov-2024	Monday	Exam Break	0
19-Nov-2024	Tuesday	Exam Break	0
20-Nov-2024	Wednesday	Exam Break	0
21-Nov-2024	Thursday	Exam Break	0
22-Nov-2024	Friday	Exam Break	0
23-Nov-2024	Saturday	Exam Break	0
24-Nov-2024	Sunday	Exam Break	0
25-Nov-2024	Monday	Exam Break	0
26-Nov-2024	Tuesday	Exam Break	0
27-Nov-2024	Wednesday	Studied the theory for development of lacing module and the review meeting	5
28-Nov-2024	Thursday	Development of algorithm for lacing Module	4
29-Nov-2024	Friday	Development of algorithm for lacing Module	4

30-Nov-2024	Saturday	Development of algorithm for lacing Module	4
01-Dec-2024	Sunday	Development of algorithm for lacing Module	4
02-Dec-2024	Monday	Development of algorithm for lacing Module	4
03-Dec-2024	Tuesday	Final review meeting of Lacing Module's Algorithm	4
04-Dec-2024	Wednesday	Make the last changes in Lacing Module	4
05-Dec-2024	Thursday	Final Submission of Algorithm for lacing module	4
06-Dec-2024	Friday	Started the calculation for the report generation for compression Module	4
07-Dec-2024	Saturday	calculation for the report generation for compression Module	4
08-Dec-2024	Sunday	Work meeting regarding the report generation and preparation of code for the same (buckling class)	4
09-Dec-2024	Monday	Work meeting regarding the report generation and preparation of code for the same (section classification)	4
10-Dec-2024	Tuesday	Work meeting regarding the report generation and preparation of code for the same (Design Checks)	4
11-Dec-2024	Wednesday	Work meeting regarding the report generation	4
12-Dec-2024	Thursday	Worked for the debugging of save$_{r}eportfunction$	5
13-Dec-2024	Friday	Worked for the debugging of save$_{r}eportfunction$	4
14-Dec-2024	Saturday	Held review meeting with mentor to discuss progress and future tasks	4
16-Dec-2024	Monday	Studied the purlin module	4
17-Dec-2024	Tuesday	First meeting regarding the development of purlin module code	4
18-Dec-2024	Wednesday	Worked for the development of purlin module	4
19-Dec-2024	Thursday	Worked for the development of purlin module	4
20-Dec-2024	Friday	Worked for the development of purlin module	4
21-Dec-2024	Saturday	Started working for the documentation of report generation	4
22-Dec-2024	Sunday	Documentation of report genration	4

23-Dec-2024	Monday	Review meeting on the report generation documnet and helped in the development of output dock for purlin module	4
24-Dec-2024	Tuesday	Completed the google doc version of the report generation document	3
25-Dec-2024	Wednesday	Made the changes suggested by mentor in the report	3
26-Dec-2024	Thursday	Final document of report generation is submitted	3
27-Dec-2024	Friday	Prepared final internship report and compiled all work done during the period	4
28-Dec-2024	Saturday	Prepared final internship report and compiled all work done during the period	3
30-Dec-2024	Monday	Made the changes in the report generation document	1

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.