FOSSEE Summer Internship Report

On

Report Generation of Steel Connection Modules for Osdag

Submitted by

Roushan Raj

2nd Year B.Tech Student, Department of Computer Science and Engineering

B.I.T. Sindri

Sindri, Dhanbad, Jharkhand

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

November 30, 2025

Acknowledgments

I would like to extend my sincere gratitude to the following individuals and organizations whose guidance and support were instrumental in the successful completion of this fellowship project:

- My heartfelt thanks go to the Osdag team, particularly **Ajmal Babu M. S.**, **Ajinkya Dahale**, and **Parth Karia**, for their continuous technical expertise and collaborative support during the report generation of modules.
- I am truly grateful for the mentorship and inspiring leadership of **Prof. Siddhartha Ghosh**, Principal Investigator of the Osdag project, from the Department of Civil Engineering at IIT Bombay.
- I would like to express my gratitude to **Prof. Kannan M. Moudgalya**, Principal Investigator of the FOSSEE project, and the entire FOSSEE team in the Department of Chemical Engineering at IIT Bombay for their invaluable project guidance and academic support.
- My sincere appreciation is extended to **Usha Viswanathan, Vineeta Parmar**, and the FOSSEE management team for their exceptional administrative support, which ensured the seamless progression of this internship.
- I gratefully acknowledge the financial and institutional support from the **National Mission on Education through Information and Communication Technology (NMEICT)**, Ministry of Education, Government of India.
- Finally, I would like to thank my alma mater, **B.I.T. Sindri**, along with the mentors and administrative staff who have supported me throughout my academic journey.

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 Report Generation of Bolted Lap Joint **9**

2.1 Task 2: Problem Statement 9

2.2 Task 2: Tasks Done 9

2.2.1 Core Save Design Function Development 9

2.2.2 Detailed Design Verification and Check Implementation 10

2.2.3 Advanced LaTeX Mathematical Formatting 11

2.2.4 Design Calculations Integration 12

2.3 Task 1: Python Code 12

2.3.1 Explanation of the Code 22

2.4 Task 1: Documentation 23

2.5 References 23

3 Report Generation of Bolted Butt Joint **24**

3.1 Task 1: Problem Statement 24

3.2 Task 1: Tasks Done 24

3.2.1 Core Save Design Function Development 24

3.2.2 Detailed Design Verification and Check Implementation 26

3.2.3 Advanced LaTeX Mathematical Formatting 27

3.2.4 Design Calculations Integration 28

3.3 Task 2: Python Code 28

3.3.1 Explanation of the Code 38

3.4 Task 2: Documentation 39
3.5 References 40

4 Report Generation of Welded Lap Joint 41
4.1 Task 3: Problem Statement 41
4.2 Task 3: Tasks Done 41
4.2.1 Development of `save_design()` Function 41
4.2.2 Check Implementation and Output 42
4.3 Task 3: Python Code 43
4.3.1 Explanation of the Code 46
4.4 Task 3: Documentation 47
4.5 References 48

5 Report Generation of Welded Butt Joint 49
5.1 Task 4: Problem Statement 49
5.2 Task 4: Tasks Done 49
5.2.1 Design Automation Workflow 49
5.2.2 Core Design Functions 50
5.2.3 Report Generation with `save_design` 51
5.3 Task 4: Python Code 52
5.3.1 Explanation of the Code 59
5.4 Task 4: Documentation 60
5.5 References 61

6 Conclusions 62
6.1 Tasks Accomplished 62
6.2 Skills Developed 62

A Appendix 66
A.1 Work Reports 66

Bibliography 72

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.

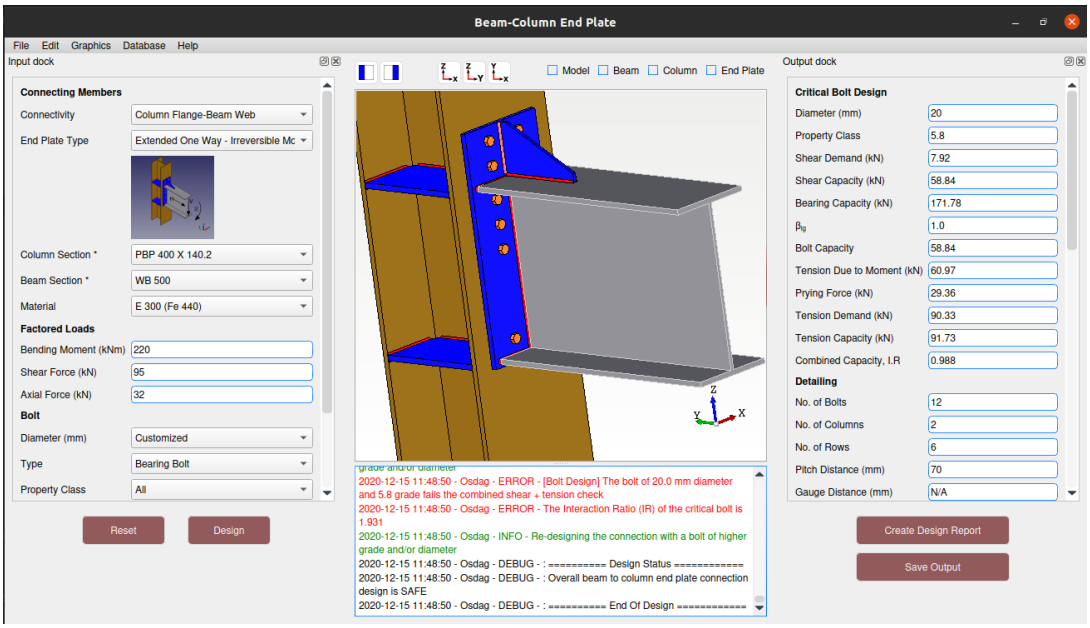

Figure 1.2: Osdag GUI

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

Report Generation of Bolted Lap Joint

2.1 Task 2: Problem Statement

The primary objective of this task was to generate a comprehensive design report for **Bolted Lap Joint Connections**. Osdag (Open Steel Design and Graphics) is an open-source tool designed for the design and detailing of steel structures in accordance with Indian Standards. A critical requirement for steel structures is ensuring they pass all Design and Detailing Checklists (DDCL) to guarantee durability, and such comprehensive design reports provide this essential information. The challenge involved meticulously converting all necessary steps, formulas, and compliance checks stipulated by IS 800:2007 into code to ensure successful report generation for bolted connections.

2.2 Task 2: Tasks Done

2.2.1 Core Save Design Function Development

- **Implemented comprehensive `save_design()` function in `lap_joint_bolted.py` with robust error handling and validation**

  Developed the main orchestrating function that serves as the entry point for report generation. This function coordinates data collection from the UI, validates all design parameters (such as plate thickness, bolt grade, and load values), performs calculations, and triggers the LaTeX report compilation. It includes comprehensive error handling to manage exceptions during file operations and data processing.

- **Integrated with existing Osdag LaTeX report infrastructure using `CreateLatex` class**

  Successfully interfaced the new bolted lap joint module with Osdag's established LaTeX report generation framework. This required constructing a detailed `report_input` dictionary and a structured `report_check` list to pass data dynamically to the `CreateLatex.save_latex` method, ensuring consistency with other connection modules.

- **Implemented proper file path handling with OS-independent path management**

  Coded cross-platform file path management using Python's `os.path` module to ensure the report generation works seamlessly across different operating systems. The function automatically handles directory creation for reports if they do not exist.

- **Report Input Dictionary Structure**

  Architected a complete data structure that captures every piece of information needed for report generation, including:

  - Module identification (Lap Joint Bolted Connection)
  - Material properties (Ultimate and Yield stress)
  - Geometric parameters (Plate thicknesses, width)
  - Bolt specifics (Diameter, Grade, Type - Friction/Bearing)
  - Load information (Tensile/Compression Force)

2.2.2 Detailed Design Verification and Check Implementation

Bolt Capacity Checks

- **Implemented Bolt Shear and Bearing calculations**

  Coded the mathematical logic for determining bolt capacity based on IS 800:2007 requirements [Cl. 10.3.3, 10.3.4]. The function dynamically calculates capacity based on the bolt type (Bearing or Friction Grip).

  Shear Capacity:

$$V_{dsb} = \frac{f_{ub} \times A_{sb}}{\gamma_{mb}} \tag{2.1}$$

Bearing Capacity:

$$V_{dpb} = \frac{2.5 \times k_b \times d \times t \times f_u}{\gamma_{mb}} \tag{2.2}$$

- **Bolt Design Capacity Logic**

  Implemented logic to determine the governing bolt value (V_{db}) by taking the minimum of shear and bearing capacities for bearing bolts, or slip resistance for friction bolts.

Base Metal Strength Analysis

- **Implemented dual strength checks (Yielding and Rupture)**

  Developed parallel calculation paths for both failure modes in tension member design [Cl. 6.2, 6.3]:

  Gross Yielding:

$$T_{dg} = \frac{A_g \times f_y}{\gamma_{m0}} \tag{2.3}$$

  Net Rupture:

$$T_{dn} = \frac{0.9 \times A_n \times f_u}{\gamma_{m1}} \tag{2.4}$$

- **Compression Capacity Check**

  Added conditional logic to handle compression loads [Cl. 7.1.2], ensuring the report adapts dynamically to the `design_for` parameter:

$$P_d = \frac{A_g \times f_y}{\gamma_{m0}} \tag{2.5}$$

2.2.3 Advanced LaTeX Mathematical Formatting

Mathematical Equation Integration

- **Used `NoEscape` LaTeX formatting for complex mathematical expressions**

  Mastered the `pylatex.utils.NoEscape` functionality to render complex mathematical equations without Python string escaping issues. This was crucial for rendering fractions, subscripts, and Greek letters (e.g., γ_{mb}, π) correctly in the final PDF.

- **Dynamic Equation Generation**

  Implemented formatted strings (f-strings) within the LaTeX code to inject calculated Python variables (like `bolt_shear_kN`, `kb`, `An`) directly into the equation display strings. This ensures the report shows the actual numerical substitution steps, not just the final result.

2.2.4 Design Calculations Integration

Utilization Ratio and Detailing

- **Calculated Utilization Ratio (UR)**

  Implemented a tracking system to calculate the utilization ratio for both the bolts and the base metal. The code identifies the critical UR to determine the overall safety of the connection:

$$UR = \frac{\text{Applied Force}}{\text{Design Capacity}} \tag{2.6}$$

- **Detailing Requirements Implementation**

  Implemented practical construction detailing checks as per IS 800:2007 [Cl. 10.2], verifying:

  – Minimum Pitch and Gauge ($2.5 \times d$)

  – Minimum and Maximum Edge Distances

  The `save_design` function automatically verifies if provided spacing (p_{prov}, e_{prov}) meets the calculated limits (p_{min}, e_{min}) and reports a "PASS" or "FAIL" status in the final document.

2.3 Task 1: Python Code

The entire logic for the report generation of Welded Butt Joint module is within the save_design() function. And it's being connected to the main GUI through ui_template.py and ui_popup_summary.py files. Below are key snippets from the module with explanations of their functionality.

Description of the Script

The script defines the save_design() function, which handles everything from user input to final design report generation. It uses reportGenerator_latex.py to manage properties and perform calculations.

Python Code

The following snippets showcase the core logic implemented in the report generation task.

Listing 2.1: Bolted Lap Joint save_design() function

```
%--------------------begin code-------------
def save_design(self, popup_summary):
        """
        Generate the LaTeX design report for Lap Joint Bolted
           Connection (Tension/Compression)
        per IS 800:2007.
        """
        import os
        from pylatex.utils import NoEscape

        try:
            # Helper functions
            def g(attr, default=None):
                v = getattr(self, attr, default)
                return default if v is None else v

            def f2(x, default=0.0):
                try:
                    return round(float(x), 2)
                except (TypeError, ValueError):
                    return default

            def as_int(x, default=0):
                try:
                    return int(round(float(x)))
                except (TypeError, ValueError):
                    return default

```

```
28              # Design status check
29              if not getattr(self, 'design_status', False):
30                  self.report_input = {
31                      KEY_MODULE: "Lap Joint Bolted",
32                      KEY_MAIN_MODULE: "Lap Joint Bolted Connection",
33                      "Design Status": "TITLE",
34                      "Status": "Design not completed successfully."
35                  }
36                  self.report_check = []
37                  self.report_check.append([
38                      "SubSection", "Design Status", "|p{2.5cm}|p{2cm}|p
                            {8cm}|p{2.5cm}|"
39                  ])
40                  self.report_check.append(["Design", "Design not
                        completed successfully.", "", "FAIL"])
41
42                  Disp_2d_image = []
43                  Disp_3D_image = "/ResourceFiles/images/3d.png"
44                  rel_path = os.path.abspath(".").replace("\\", "/")
45
46                  fname_no_ext = popup_summary.get("filename", "
                        LapJointBoltedReport")
47                  folder = popup_summary.get('folder', './reports')
48                  os.makedirs(folder, exist_ok=True)
49
50                  CreateLatex.save_latex(
51                      CreateLatex(), self.report_input, self.report_check
                            ,
52                      popup_summary, fname_no_ext, rel_path,
                            Disp_2d_image, Disp_3D_image,
53                      module=getattr(self, 'module', 'Lap Joint Bolted')
54                  )
55                  return True
56
57              # Extract design values
58              self.module = g('module', 'Lap Joint Bolted')
59              self.mainmodule = 'Lap Joint Bolted Connection'
60              design_for = str(g('design_for', 'Tension')).strip()
61              is_comp = design_for.lower().startswith('c')
```

```
62
63              # Geometry
64              plate1_thk = f2(g('plate1thk', g('pltthk', 0.0)), 0.0)
65              plate2_thk = f2(g('plate2thk', g('pltthk', 0.0)), 0.0)
66              width = f2(g('width', 0.0), 0.0)
67
68              # Forces
69              axial_kN = f2(g('axial_force_kN', g('tensile_force', 0.0)),
                    0.0)
70
71              # Bolts
72              bolt_dia_prov = f2(getattr(self.bolt, '
                   bolt_diameter_provided', 0.0) if hasattr(self, 'bolt')
                   else 0.0, 0.0)
73              bolt_grade_prov = f2(getattr(self.bolt, '
                   bolt_grade_provided', 0.0) if hasattr(self, 'bolt') else
                    0.0, 0.0)
74              bolt_type = getattr(self.bolt, 'bolt_type',
                   VALUE_NOT_APPLICABLE) if hasattr(self, 'bolt') else
                   VALUE_NOT_APPLICABLE
75
76              bolt_shear_kN = f2(getattr(self.bolt, 'bolt_shear_capacity'
                   , 0.0) if hasattr(self, 'bolt') else 0.0, 0.0)
77              bolt_bearing_kN = f2(getattr(self.bolt, '
                   bolt_bearing_capacity', 0.0) if hasattr(self, 'bolt')
                   else 0.0, 0.0)
78              bolt_final_cap = f2(getattr(self.bolt, 'bolt_capacity',
                   0.0) if hasattr(self, 'bolt') else 0.0, 0.0)
79
80              # Layout
81              rows = as_int(g('rows', 0), 0)
82              cols = as_int(g('cols', 0), 0)
83              n_bolts = as_int(g('number_bolts', 0), 0)
84              pitch = as_int(g('final_pitch', 0), 0)
85              gauge = f2(g('final_gauge', 0.0), 0.0)
86              e_dist = f2(g('final_edge_dist', 0.0), 0.0)
87
88              # Spacing limits
89              p_min = as_int(getattr(self.bolt, 'min_pitch', 0) if
```

```
            hasattr(self, 'bolt') else 0, 0)
        g_min = as_int(getattr(self.bolt, 'min_gauge', 0) if
            hasattr(self, 'bolt') else 0, 0)
        e_min = f2(getattr(self.bolt, 'min_edge_dist', 0.0) if
            hasattr(self, 'bolt') else 0.0, 0.0)
        e_max = f2(getattr(self.bolt, 'max_edge_dist', 0.0) if
            hasattr(self, 'bolt') else 0.0, 0.0)

        # Material properties
        t_fu_fy_list = getattr(self, 'bolt_conn_plates_t_fu_fy',
            [])
        if t_fu_fy_list and len(t_fu_fy_list) > 0:
            fu = t_fu_fy_list[0][1] if len(t_fu_fy_list[0]) > 1
                else 0
            fy = t_fu_fy_list[0][2] if len(t_fu_fy_list[0]) > 2
                else 0
        else:
            fy = g('yield_stress', 0)
            fu = 0

        # Base metal
        base_metal_capacity_kN = f2(g('base_metal_capacity_kN',
            0.0), 0.0)
        A_g = f2(g('A_g', 0.0), 0.0)
        Tdg = f2(g('Tdg', 0.0), 0.0)
        Tdn = f2(g('Tdn', 0.0), 0.0)
        T_db = f2(g('T_db', 0.0), 0.0)

        # Utilization
        overall_ur = round(g('utilization_ratio', 0.0), 3)

        # Bolt values for equations
        bolt_fu = as_int(bolt_grade_prov * 100, 0)
        bolt_net_area = f2(math.pi * (bolt_dia_prov ** 2) / 4, 0.0)
        kb = f2(getattr(self.bolt, 'kb', 1.0) if hasattr(self, '
            bolt') else 1.0, 1.0)

        # Build report input
        self.report_input = {
```

```
                KEY_MODULE: "Lap Joint Bolted Connection",
                KEY_DISP_DESIGN_FOR: design_for,
                "Material *": getattr(self, 'main_material',
                    VALUE_NOT_APPLICABLE),
                "Thickness of Plate-1 (mm) *": plate1_thk,
                "Thickness of Plate-2 (mm) *": plate2_thk,
                "Width of Plate (mm) *": width,
                f"{'Tensile' if not is_comp else 'Axial'} Force (kN) *"
                    : axial_kN,
                "Diameter (mm) *": bolt_dia_prov,
                "Property Class *": bolt_grade_prov,
                "Type *": bolt_type,
            }

            # Build report check - FIXED COLUMN WIDTHS
            self.report_check = []

            # Section 2.1: Bolt Capacity - NARROW "Required" column
            self.report_check.append([
                "SubSection", "Bolt Capacity Calculations", "|p{2.5cm}|
                    p{2cm}|p{8cm}|p{2.5cm}|"
            ])

            if bolt_type == TYP_BEARING:
                shear_eq = NoEscape(
                    r"$V_{dsb} = \frac{f_{ub} \times A_{sb}}{\gamma_{mb
                        }}$" + "\n\n" +
                    rf"$= \frac{{{bolt_fu} \times \pi \times {
                        bolt_dia_prov}^2/4}}{{1.25}}$" + "\n\n" +
                    rf"$= {bolt_shear_kN}$ kN" + "\n\n" +
                    r"[Ref: Cl. 10.3.3, IS 800:2007]"
                )
                self.report_check.append(["Bolt Shear Capacity", "",
                    shear_eq, ""])

                t_plate = min(plate1_thk, plate2_thk) if plate1_thk > 0
                    and plate2_thk > 0 else plate1_thk
                bearing_eq = NoEscape(
                    r"$V_{dpb} = \frac{2.5 \times k_b \times d \times t
```

```
                        \times f_u}{\gamma_{mb}}$" + "\n\n" +
                rf"$= \frac{{2.5 \times {kb} \times {bolt_dia_prov}
                    \times {t_plate} \times {fu}}}{{1.25}}$" + "\n\
                    n" +
                rf"$= {bolt_bearing_kN}$ kN" + "\n\n" +
                r"[Ref: Cl. 10.3.4, IS 800:2007]"
            )
            self.report_check.append(["Bolt Bearing Capacity", "",
                bearing_eq, ""])

        bolt_cap_eq = NoEscape(
            r"$V_{db} = \min(V_{dsb}, V_{dpb})$" + "\n\n" +
            rf"$= \min({bolt_shear_kN}, {bolt_bearing_kN})$" + "\n\
                n" +
            rf"$= {bolt_final_cap}$ kN"
        )
        self.report_check.append(["Bolt Design Capacity", "",
            bolt_cap_eq, ""])

        # Section 2.2: Number of Bolts
        self.report_check.append([
            "SubSection", "Number of Bolts Required", "|p{2.5cm}|p
                {2cm}|p{8cm}|p{2.5cm}|"
        ])

        bolts_req = as_int(axial_kN / bolt_final_cap if
            bolt_final_cap > 0 else 0, 0)
        bolts_eq = NoEscape(
            r"$n = \frac{P}{V_{db}}$" + "\n\n" +
            rf"$= \frac{{{axial_kN}}}{{{bolt_final_cap}}}$" + "\n\n
                " +
            rf"$= {bolts_req}$ nos."
        )
        self.report_check.append(["Bolts Required", "", bolts_eq, "
            "])

        # Section 2.3: Bolt Arrangement
        self.report_check.append([
            "SubSection", "Bolt Arrangement", "|p{2.5cm}|p{2cm}|p{8
```

```
                cm}|p{2.5cm}|"
            ])

            self.report_check.append([
                "Bolt Pattern", "", f"Arrangement: {rows} rows    {cols
                } columns", ""
            ])

            # Section 2.4: Detailing Requirements
            self.report_check.append([
                "SubSection", "Detailing Requirements", "|p{2.5cm}|p{4
                cm}|p{6cm}|p{2.5cm}|"
            ])

            spacing_req = NoEscape(
                rf"$p_{{\min}} = 2.5d = {p_min}$ mm" + "\n\n" +
                rf"$g_{{\min}} = 2.5d = {g_min}$ mm" + "\n\n" +
                r"[Cl. 10.2.2]"
            )
            spacing_prov = NoEscape(
                rf"$p_{{prov}} = {pitch}$ mm" + "\n\n" +
                rf"$g_{{prov}} = {gauge}$ mm"
            )
            self.report_check.append(["Minimum Spacing", spacing_req,
               spacing_prov, "PASS"])

            edge_req = NoEscape(
                rf"$e_{{\min}} = {e_min}$ mm" + "\n\n" +
                rf"$e_{{\max}} = {e_max}$ mm" + "\n\n" +
                r"[Cl. 10.2.4]"
            )
            edge_prov = NoEscape(rf"$e_{{prov}} = {e_dist}$ mm")
            self.report_check.append(["Edge Distance", edge_req,
               edge_prov, "PASS"])

            # Section 2.5: Base Metal Strength
            self.report_check.append([
                "SubSection", "Base Metal Strength", "|p{3cm}|p{2cm}|p
                {7cm}|p{2.5cm}|"
```

```
            ])

            if is_comp:
                comp_eq = NoEscape(
                    r"$P_d = \frac{A_g \times f_y}{\gamma_{m0}}$" + "\n\n" +
                    rf"$= \frac{{{A_g:.1f} \times {fy}}}{{1.1}} = {base_metal_capacity_kN:.2f}$ kN" + "\n\n" +
                    r"[Cl. 7.1.2]"
                )
                self.report_check.append([
                    "Compression", "", comp_eq, "PASS" if base_metal_capacity_kN >= axial_kN else "FAIL"
                ])
            else:
                if Tdg > 0 and Tdn > 0:
                    ten_eq = NoEscape(
                        rf"$T_{{dg}} = {Tdg:.2f}$ kN (Gross yielding)" + "\n\n" +
                        rf"$T_{{dn}} = {Tdn:.2f}$ kN (Net rupture)" + "\n\n" +
                        rf"$T_d = {T_db:.2f}$ kN" + "\n\n" +
                        r"[Cl. 6.2, 6.3]"
                    )
                else:
                    ten_eq = NoEscape(rf"$T_d = {base_metal_capacity_kN:.2f}$ kN" + "\n\n" + r"[Cl. 6.2/6.3]")

                self.report_check.append([
                    "Tension", "", ten_eq, "PASS" if base_metal_capacity_kN >= axial_kN else "FAIL"
                ])

            # Section 2.6: Design Summary
            self.report_check.append([
                "SubSection", "Design Summary", "|p{2.5cm}|p{2cm}|p{8cm}|p{2.5cm}|"
            ])

```

```
            total_capacity = f2(bolt_final_cap * n_bolts, 0.0)
            ur_eq = NoEscape(
                r"$UR = \frac{\text{Applied Force}}{\text{Design Capacity}}$" + "\n\n" +
                rf"$= \frac{{{axial_kN}}}{{{total_capacity}}} = {overall_ur}$"
            )
            self.report_check.append(["Utilization Ratio", "", ur_eq, ""])

            design_status_str = "Design is SAFE" if overall_ur <= 1.0 else "Design is UNSAFE"
            self.report_check.append([
                "Design Status", "", design_status_str, "PASS" if overall_ur <= 1.0 else "FAIL"
            ])

            Disp_2d_image = []
            Disp_3D_image = "/ResourceFiles/images/3d.png"
            rel_path = os.path.abspath(".").replace("\\", "/")
            fname_no_ext = popup_summary.get("filename", "LapJointBoltedReport")
            folder = popup_summary.get('folder', './reports')
            os.makedirs(folder, exist_ok=True)

            CreateLatex.save_latex(
                CreateLatex(), self.report_input, self.report_check,
                popup_summary, fname_no_ext, rel_path, Disp_2d_image, Disp_3D_image,
                module=self.module
            )
            logger.info(f"Report generated successfully: {fname_no_ext}.pdf")
            return True

        except Exception as e:
            print(f"CRITICAL ERROR in save_design(): {e}")
            return False
%-------------------- end code ---------------
```

2.3.1 Explanation of the Code

The `save_design` function in the Bolted Lap Joint module serves as the core automation engine for generating the final design report. It orchestrates the entire process from data retrieval to PDF generation. The function's key operations are summarized below:

1. **Data Retrieval and Initialization:** The function first extracts the design parameters stored in the class instance. This includes the tensile or compressive load (P), plate dimensions (t_1, t_2, w), and bolt properties (diameter d, grade, and hole type). It also identifies the connection mode (Bearing or Friction Grip) to apply the correct design standards.

2. **Design Status Verification:** A critical check is performed on the `design_status` attribute.

   - If the design is marked as **False** (failed), the function generates a brief report summarizing the failure (e.g., "Bolt capacity zero" or "Plate utilization ¿ 1.0").
   - If **True** (safe), it proceeds to compile the full report with all calculation details.

3. **Input Dictionary Construction:** The function builds the `report_input` dictionary, which acts as the data bridge to the LaTeX template. Key entries include:

   - **Geometry:** Thickness of connected plates and overlap dimensions.
   - **Bolt Data:** Shear capacity (V_{dsb}) and bearing capacity (V_{dpb}) per bolt.
   - **Arrangement:** Number of rows, columns, pitch (p), and gauge (g).

4. **Automated LaTeX Code Generation:** Using the `pylatex` library, the function dynamically creates LaTeX equations. It injects Python variables directly into math strings to show the step-by-step verification:

   - **Shear Check:** $V_{dsb} = \frac{f_u \times A_s}{\sqrt{3} \times \gamma_{mb}}$
   - **Bearing Check:** $V_{dpb} = 2.5 \times k_b \times d \times t \times f_u$
   - **Utilization Ratio:** $UR = \frac{F_{applied}}{F_{capacity}}$

This ensures the report reflects the exact state of the current design iteration.

5. **Report Compilation and Export:** Finally, the function calls `CreateLatex.save_latex` to render the complete document. It handles file paths robustly using `os.path` to ensure cross-platform compatibility and automatically embeds 2D/3D images of the joint configuration.

2.4 Task 1: Documentation

The directory structure used for the Report Generation is organized as follows:

```
src/osdag

 design_report/
    reportGenerator_latex.py

 design_type/
    connection/
       lap_joint_bolted.py

 gui
    ui_summary_popup.py/
    ui_template.py/
Report_functions.py
```

Program Start Calls The main entry point for the program is osdagMainPage.py . To start the program, open the Osdag folder and start the terminal with that path, execute the following command from the project root:

```
$ python -m osdag.osdagMainPage.py
```

2.5 References

- Osdag Github repository: https://osdag.github.io/
- IS 800:2007, "General Construction in Steel Code of Practice of Practice"

Chapter 3

Report Generation of Bolted Butt Joint

3.1 Task 1: Problem Statement

The primary objective of this task was to generate a comprehensive design report for **Bolted Butt Joint Connections**. Osdag (Open Steel Design and Graphics) is an open-source tool designed for the design and detailing of steel structures in accordance with Indian Standards. A critical requirement for steel structures is ensuring they pass all Design and Detailing Checklists (DDCL) to guarantee durability and safety. The challenge involved meticulously converting all necessary steps, formulas, and compliance checks stipulated by IS 800:2007 into code to ensure successful report generation for butt joint connections, which involve complex interactions between main plates, cover plates, and bolts.

3.2 Task 1: Tasks Done

3.2.1 Core Save Design Function Development

- **Implemented comprehensive `save_design()` function in `butt_joint_bolted.py` with robust error handling**

  Developed the main orchestrating function that serves as the entry point for

report generation. This function coordinates data collection from the UI, validates all design parameters (including cover plate thickness and packing plates), performs calculations, and triggers the LaTeX report compilation. It includes comprehensive error handling to manage exceptions during file operations and data processing.

- **Integrated with existing Osdag LaTeX report infrastructure using `CreateLatex` class**

  Successfully interfaced the new bolted butt joint module with Osdag's established LaTeX report generation framework. This required constructing a detailed `report_input` dictionary and a structured `report_check` list to pass data dynamically to the `CreateLatex.save_latex` method, ensuring consistency with other connection modules.

- **Implemented proper file path handling with OS-independent path management**

  Coded cross-platform file path management using Python's `os.path` module to ensure the report generation works seamlessly across different operating systems. The function automatically handles directory creation for reports if they do not exist.

- **Report Input Dictionary Structure**

  Architected a complete data structure that captures every piece of information needed for report generation, including:

  * Module identification (Butt Joint Bolted Connection)
  * Material properties (Ultimate and Yield stress)
  * Geometric parameters (Main plate thickness, Cover plate thickness)
  * Bolt specifics (Diameter, Grade, Type - Friction/Bearing)
  * Load information (Tensile/Compression Force)
  * Cover Plate details (Single/Double, Thickness)

3.2.2 Detailed Design Verification and Check Implementation

Cover Plate and Packing Plate Checks

– **Implemented Cover Plate Thickness Logic**

Coded logic to automatically determine required cover plate thickness based on the main plate thickness (T_{min}) and connection type (Single/Double Cover) [Cl. 10.3.3.3]:

$$T_{cp} \geq \begin{cases} \frac{5}{8}T_{min} & \text{for Single Cover} \\ \frac{9}{16}T_{min} & \text{for Double Cover} \end{cases} \tag{3.1}$$

– **Packing Plate and Reduction Factor (β_{pkg})**

Implemented checks for packing plates when connecting plates of different thicknesses. If the packing thickness exceeds 6mm, a reduction factor (β_{pkg}) is applied to the bolt shear capacity [Cl. 10.3.3.3]:

$$\beta_{pkg} = 1 - 0.0125 \times t_{pkg} \tag{3.2}$$

Bolt Capacity Checks

– **Implemented Bolt Shear and Bearing calculations**

Coded the mathematical logic for determining bolt capacity based on IS 800:2007 requirements [Cl. 10.3.3, 10.3.4]. The function dynamically calculates capacity based on the bolt type (Bearing or Friction Grip).

Shear Capacity:

$$V_{dsb} = \frac{f_{ub} \times A_{sb}}{\gamma_{mb}} \tag{3.3}$$

Bearing Capacity:

$$V_{dpb} = \frac{2.5 \times k_b \times d \times t \times f_u}{\gamma_{mb}} \tag{3.4}$$

– **Capacity Reduction Factors**

Implemented advanced checks for "Long Joints" (β_{lj}) and "Large Grips" (β_{lg})

as per Cl. 10.3.3.1 and 10.3.3.2. The code automatically reduces bolt capacity if the joint length or grip length exceeds specified limits:

$$\beta_{lj} = 1.075 - \frac{l_j}{200d} \quad (0.75 \leq \beta_{lj} \leq 1.0) \tag{3.5}$$

Base Metal Strength Analysis

- **Implemented dual strength checks (Yielding and Rupture)**

  Developed parallel calculation paths for both failure modes in tension member design [Cl. 6.2, 6.3]:

  Gross Yielding:

  $$T_{dg} = \frac{A_g \times f_y}{\gamma_{m0}} \tag{3.6}$$

  Net Rupture:

  $$T_{dn} = \frac{0.9 \times A_n \times f_u}{\gamma_{m1}} \tag{3.7}$$

- **Block Shear Check**

  Integrated Block Shear Failure checks (V_{db}) [Cl. 6.4] to ensure the connection does not fail by tearing out a block of material. The final design strength (T_{db}) is taken as the minimum of yielding, rupture, and block shear capacities.

3.2.3 Advanced LaTeX Mathematical Formatting

Mathematical Equation Integration

- **Used `NoEscape` LaTeX formatting for complex mathematical expressions**

  Mastered the `pylatex.utils.NoEscape` functionality to render complex mathematical equations without Python string escaping issues. This was crucial for rendering fractions, subscripts, and Greek letters (e.g., γ_{mb}, π) correctly in the final PDF.

- **Dynamic Equation Generation**

  Implemented formatted strings (f-strings) within the LaTeX code to inject calculated Python variables (like `bolt_shear_kN`, `kb`, `An`) directly into the

equation display strings. This ensures the report shows the actual numerical substitution steps, not just the final result.

3.2.4 Design Calculations Integration

Utilization Ratio and Detailing

- **Calculated Utilization Ratio (UR)**

  Implemented a tracking system to calculate the utilization ratio for both the bolts and the base metal. The code identifies the critical UR to determine the overall safety of the connection:

$$UR = \frac{\text{Applied Force}}{\text{Design Capacity}} \tag{3.8}$$

- **Detailing Requirements Implementation**

  Implemented practical construction detailing checks as per IS 800:2007 [Cl. 10.2], verifying:

  * Minimum Pitch and Gauge ($2.5 \times d$)
  * Minimum and Maximum Edge Distances

  The `save_design` function automatically verifies if provided spacing (p_{prov}, e_{prov}) meets the calculated limits (p_{min}, e_{min}) and reports a "PASS" or "FAIL" status in the final document.

3.3 Task 2: Python Code

The entire logic for the report generation of Bolted Butt Joint module is within the save_design() function. And it's being connected to the main GUI through ui_template.py and ui_popup_summary.py files. Below are key snippets from the module with explanations of their functionality.

Description of the Script

The script defines the save_design() function, which handles everything from user input to final design report generation. It uses reportGenerator_latex.py to manage properties and perform calculations.

Python Code

The following snippets showcase the core logic implemented in the report generation task.

Listing 3.1: Bolted Butt Joint save_design() function

```
%--------------------begin code-------------
def save_design(self, popup_summary):
        """
        Generate the LaTeX design report for Bolted Butt Joint (
            Tension or Compression) per IS 800:2007.
        """
        try:
            # ---------- INPUTS BLOCK ----------
            self.report_input = {
                KEY_MODULE: getattr(self, 'module', 'Butt Joint
                    Bolted'),
                KEY_MAIN_MODULE: getattr(self, 'mainmodule', 'Butt
                    Joint Bolted Connection'),

                # Show design intent (Tension/Compression) and
                    axial force (kN)
                KEY_DISP_DESIGN_FOR: getattr(self, 'design_for', '
                    Tension'),
                KEY_DISP_AXIAL_FORCE: float(getattr(self, '
                    axial_force_kN', getattr(self, 'tensile_force',
                    0.0))),  # kN

                # Connection details
                "Connection Details": "TITLE",
                KEY_DISP_MATERIAL: getattr(self, 'main_material', '
                    N/A'),
```

```
19                  KEY_DISP_PLATE1_THICKNESS: float(getattr(self, '
                        plate1thk', 0)),
20                  KEY_DISP_PLATE2_THICKNESS: float(getattr(self, '
                        plate2thk', 0)),
21                  KEY_DISP_PLATE_WIDTH: float(getattr(self, 'width',
                        0)),
22                  KEY_DISP_COVER_PLATE: getattr(self, 'cover_plate',
                        'Both Sides'),
23
24                  # Material properties (from plate1 as
                        representative)
25                  "Material Properties": "TITLE",
26                  KEY_DISP_ULTIMATE_STRENGTH_REPORT: round(getattr(
                        self.plate1, 'fu', 0), 1) if hasattr(self, '
                        plate1') else 0,
27                  KEY_DISP_YIELD_STRENGTH_REPORT: round(getattr(self.
                        plate1, 'fy', 0), 1) if hasattr(self, 'plate1')
                        else 0,
28
29                  # Bolt details + design preferences (full lists
                        like tension-bolted module)
30                  "Bolt Details - Input and Design Preference": "
                        TITLE",
31                  KEY_DISP_D: str([int(d) for d in getattr(self.bolt,
                         'bolt_diameter', [])]) if hasattr(self, 'bolt')
                         else "[]",
32                  KEY_DISP_GRD: str([float(d) for d in getattr(self.
                        bolt, 'bolt_grade', [])]) if hasattr(self, 'bolt
                        ') else "[]",
33                  KEY_DISP_TYP: getattr(self.bolt, 'bolt_type', 'N/A'
                        ) if hasattr(self, 'bolt') else 'N/A',
34                  KEY_DISP_DP_BOLT_HOLE_TYPE: getattr(self.bolt, '
                        bolt_hole_type', 'Standard') if hasattr(self, '
                        bolt') else 'Standard',
35
36                  "Detailing - Design Preference": "TITLE",
37                  KEY_DISP_DP_DETAILING_EDGE_TYPE: getattr(self.bolt,
                         'edge_type', 'Sheared or hand flame cut') if
                        hasattr(self, 'bolt') else 'Sheared or hand
```

```
                    flame cut',
38              KEY_DISP_DP_DETAILING_CORROSIVE_INFLUENCES_BEAM:
                    getattr(self.bolt, 'corrosive_influences', '
                    Corrosive') if hasattr(self, 'bolt') else '
                    Corrosive',
39          }
40          if hasattr(self, 'bolt') and getattr(self.bolt, '
                bolt_type', '') == TYP_FRICTION_GRIP:
41              self.report_input[
                    KEY_DISP_DP_BOLT_SLIP_FACTOR_REPORT] = getattr(
                    self.bolt, 'mu_f', 0.3)
42
43          # ---------- CHECKS BLOCK ----------
44          self.report_check = []
45
46          if getattr(self, 'design_status', False):
47              # --- Section: Spacing Checks (Cl. 10.2) ---
48              self.report_check.append(('SubSection', 'Spacing
                    Check as per Cl. 10.2 of IS 800:2007',
49                                        '|p{2.5cm}|p{7.5cm}|p{3cm}|
                                            p{2.5cm}|'))
50
51              bolt_d = float(getattr(self.bolt, '
                    bolt_diameter_provided', 0)) if hasattr(self, '
                    bolt') else 0.0
52              plates_t = [getattr(self, 'plate1thk', 0), getattr(
                    self, 'plate2thk', 0)]
53              p = float(getattr(self, 'final_pitch', 0))
54              g = float(getattr(self, 'final_gauge', 0))
55              e_edge = float(getattr(self, 'final_edge_dist', 0))
56              e_end = float(getattr(self, 'final_end_dist', 0))
57              edge_type_str = getattr(self.bolt, 'edge_type', '
                    Sheared or hand flame cut') if hasattr(self, '
                    bolt') else 'Sheared or hand flame cut'
58
59              self.report_check.append((DISP_MIN_PITCH,
                    cl_10_2_2_min_spacing(bolt_d),
60                                        display_prov(p, "p", "mm"),
61                                        get_pass_fail(2.5 * bolt_d,
```

```
                                        p, relation='leq')))
62              self.report_check.append((DISP_MAX_PITCH,
                    cl_10_2_3_1_max_spacing(plates_t),
63                                  display_prov(p, "p", "mm"),
64                                  get_pass_fail(p, 32 * min(
                                        plates_t), relation='leq
                                        ')))
65
66              if g > 0:
67                  self.report_check.append((DISP_MIN_GAUGE,
                        cl_10_2_2_min_spacing(bolt_d),
68                                      display_prov(g, "g", "
                                            mm"),
69                                      get_pass_fail(2.5 *
                                            bolt_d, g, relation=
                                            'leq')))
70                  self.report_check.append((DISP_MAX_GAUGE,
                        cl_10_2_3_1_max_spacing(plates_t),
71                                      display_prov(g, "g", "
                                            mm"),
72                                      get_pass_fail(g, 32 *
                                            min(plates_t),
                                            relation='leq')))
73
74              self.report_check.append((DISP_MIN_END,
                    cl_10_2_4_2_min_edge_end_dist(bolt_d,
                    edge_type_str),
75                                  display_prov(e_end, "e_{end
                                        }", "mm"),
76                                  get_pass_fail(1.2 * bolt_d,
                                        e_end, relation='leq'))
                                    )
77              self.report_check.append((DISP_MIN_EDGE,
                    cl_10_2_4_2_min_edge_end_dist(bolt_d,
                    edge_type_str),
78                                  display_prov(e_edge, "e", "
                                        mm"),
79                                  get_pass_fail(1.2 * bolt_d,
                                        e_edge, relation='leq')
```

```
                                        ))
80
81                  # --- Section: Bolt Design (Cl. 10.3 / 10.4) ---
82                  self.report_check.append(('SubSection', 'Bolt
                        Design as per Cl. 10.3 of IS 800:2007',
83                                            '|p{2.5cm}|p{5.5cm}|p{6.5cm
                                                }|p{1cm}|'))
84
85                  bolt_type = getattr(self.bolt, 'bolt_type', '') if
                        hasattr(self, 'bolt') else ''
86                  planes = getattr(self, 'planes', 1)
87                  bolt_fu = getattr(self.bolt, 'bolt_fu', 0) if
                        hasattr(self, 'bolt') else 0
88                  bolt_net_area = getattr(self.bolt, 'bolt_net_area',
                         0) if hasattr(self, 'bolt') else 0
89                  bolt_cap = getattr(self.bolt, 'bolt_capacity', 0)
                        if hasattr(self, 'bolt') else 0
90                  # convert  N kN  if incoming in N
91                  bolt_cap_kn = round(bolt_cap / 1000.0, 2) if
                        bolt_cap > 1000 else bolt_cap
92
93                  if bolt_type == TYP_BEARING:
94                      # Shear & bearing (bearing-type)     values
                            already computed in design
95                      self.report_check.append((
                            KEY_OUT_DISP_BOLT_SHEAR, '',
96                           cl_10_3_3_bolt_shear_capacity(bolt_fu,
                                planes, bolt_net_area, 1.25,
97                                                    round(getattr(
                                                        self.bolt, '
                                                        bolt_shear_capacity
                                                        ', 0)
                                                        /1000.0,2)),
                                                         ''))
98                      kb = getattr(self.bolt, 'kb', 0)
99                      t_fu_fy = getattr(self, '
                            bolt_conn_plates_t_fu_fy', [])
100                     self.report_check.append((
                            KEY_OUT_DISP_BOLT_BEARING, '',
```

```
                        cl_10_3_4_bolt_bearing_capacity(kb, bolt_d,
                            t_fu_fy, 1.25,
                                                    round(
                                                        getattr(
                                                        self.
                                                        bolt, '
                                                        bolt_bearing_capacit
                                                        ', 0)
                                                        /1000.0,2)
                                                        ), ''))
                else:
                    # HSFG slip resistance (Cl. 10.4.3)
                    mu_f = getattr(self.bolt, 'mu_f', 0.3)
                    slip_res = round(getattr(self, 'slip_res', 0)
                        /1000.0, 2) if getattr(self, 'slip_res', 0)
                        > 1000 else getattr(self, 'slip_res', 0)
                    self.report_check.append((
                        KEY_OUT_DISP_BOLT_SLIP_DR, '',
                         cl_10_4_3_HSFG_bolt_capacity(mu_f, planes,
                            1.0, bolt_fu, bolt_net_area, 1.25,
                            slip_res), ''))

                # Layout counts
                self.report_check.append((DISP_NUM_OF_BOLTS, '',
                    display_prov(int(getattr(self, 'number_bolts',
                    0)), "n"), ''))
                self.report_check.append((DISP_NUM_OF_COLUMNS, '',
                    display_prov(int(getattr(self, 'cols', 1)), "n_{
                    c}"), ''))
                self.report_check.append((DISP_NUM_OF_ROWS, '',
                    display_prov(int(getattr(self, 'rows', 1)), "n_{
                    r}"), ''))

                # Long-joint & large-grip reductions (display only
                    if applied)
                if self.number_bolts > 2 and self.rows > 1:
                    lj = (self.rows - 1) * self.bolt.
                        min_pitch_round
                    if lj > 15 * bolt_d:
```

```
                        bij = getattr(self, 'bij', 0)
                        if 0.75 <= bij <= 1.0:
                            self.report_check.append((
                                KEY_OUT_LONG_JOINT, '',
                                cl_10_3_3_1_long_joint_reduction_factor
                                    (lj, bolt_d, bij),
                                get_pass_fail(bij, 0.75, relation='
                                    geq')))
                lg = self.plate1thk + self.plate2thk
                if lg > 5 * bolt_d:
                    blg = getattr(self, 'blg', 0)
                    if blg > 0 and blg <= 1.0:
                        self.report_check.append((
                            KEY_OUT_LARGE_GRIP, '',
                            cl_10_3_3_2_large_grip_reduction_factor
                                (lg, bolt_d, blg),
                            get_pass_fail(blg, 0.0, relation='gt'))
                                )

                self.report_check.append(('SubSection', 'Base Metal
                    Strength', '|p{4cm}|p{6cm}|p{4cm}|p{2cm}|'))

                # Make sure capacities from base-metal calc are
                    available
                _ok_bm = self.check_base_metal_strength()
                force_kN = float(getattr(self, 'axial_force_kN',
                    getattr(self, 'tensile_force', 0.0)))

                if getattr(self, 'design_for', 'Tension') == '
                    Compression':
                    # Compression: Pd = Ag*fy/gamma_m0
                    Pd = self.A_g * min(self.plate1.fy, self.plate2
                        .fy) / self.gamma_m0
                    self.report_check.append((
                        'Base Metal Capacity (Compression)',
                        NoEscape(f'\\small P$_d$ = A$_g$    f$_y$/
                            $_ {{m0}}$ = {self.A_g:.1f}  {min(self.
                            plate1.fy, self.plate2.fy)}/{self.
                            gamma_m0} = {round(Pd/1000.0,2)} kN'),
```

```
                        f'{round(Pd/1000.0,2)} kN',
                        'Pass' if Pd/1000.0 >= force_kN else 'Fail'
                    ))
                    T_db = Pd
                else:
                    # Tension: show min(Tdg, Tdn, Vdb)
                    Tdg_kN = round(self.T_dg/1000.0, 2)
                    Tdn_kN = round(self.T_dn/1000.0, 2)
                    Vdb_kN = round(self.V_db/1000.0, 2)
                    Tdb_kN = round(self.T_db/1000.0, 2)
                    self.report_check.append((
                        'Base Metal Capacity (Tension)',
                        NoEscape(f'\\small min(T$_{{dg}}$, T$_{{dn}}$, V$_{{db}}$) = min({Tdg_kN}, {Tdn_kN}, {Vdb_kN}) = {Tdb_kN} kN'),
                        f'{Tdb_kN} kN',
                        'Pass' if Tdb_kN >= force_kN else 'Fail'
                    ))
                    T_db = self.T_db

                # --- Section: Connection Verification / Summary ---
                self.report_check.append(('SubSection', 'Connection Verification', '|p{2.5cm}|p{5.5cm}|p{6.5cm}|p{1cm}|'))

                # Bolt-group total capacity
                bolt_total_kN = (self.bolt.bolt_capacity or 0.0) * int(getattr(self, 'number_bolts', 0))
                # Overall connection capacity = min(bolt-group, base metal)
                overall_capacity_kN = min(bolt_total_kN, T_db / 1000.0 if T_db > 1000 else T_db)

                # Utilization
                util = 0.0
                if overall_capacity_kN > 0:
                    util = force_kN / overall_capacity_kN

```

```
                self.report_check.append((
                    'Overall Capacity',
                    NoEscape(f'\\small min(Bolt Group, Base Metal)
                        = min({round(bolt_total_kN,2)}, {round((T_db
                        /1000.0 if T_db>1000 else T_db),2)}) = {
                        round(overall_capacity_kN,2)} kN'),
                    f'{round(overall_capacity_kN,2)} kN',
                    'Pass' if overall_capacity_kN >= force_kN else
                        'Fail'
                ))
                self.report_check.append((
                    KEY_DISP_UTILIZATION_RATIO, 'Utilization ratio
                        should be     1.0',
                    display_prov(round(util,3), "U.R."),
                    get_pass_fail(util, 1.0, relation='leq')
                ))

            else:
                self.report_check.append(('SubSection', 'Design
                    Status', '|p{2.5cm}|p{7.5cm}|p{3cm}|p{2.5cm}|'))
                self.report_check.append((
                    'Design Status',
                    'Design not completed successfully. Check input
                        parameters and design constraints.',
                    'Review inputs and try again',
                    'FAIL'
                ))

            # ---------- REPORT GENERATION ----------
            Disp_2d_image = []
            Disp_3D_image = "/ResourceFiles/images/3d.png"
            import sys, os
            rel_path = os.path.abspath(".").replace("\\", "/")
            fname_no_ext = popup_summary.get('filename', '
                Butt_Joint_Welded_Report')
            folder = popup_summary.get('folder', './reports')
            os.makedirs(folder, exist_ok=True)

            popup_summary["folder"] = folder
```

```
206             popup_summary["filename"] = fname_no_ext
207
208             CreateLatex.save_latex(CreateLatex(), self.report_input
                    , self.report_check,
209                                    popup_summary, fname_no_ext,
                                           rel_path, Disp_2d_image,
210                                    Disp_3D_image, module=self.module)
211             logger.info(f"Report generated successfully: {
                    fname_no_ext}.pdf")
212
213         except Exception as e:
214             print(f"CRITICAL ERROR in save_design(): {e}")
215             return False
216         return True
217 %-------------------- end code ---------------
```

3.3.1 Explanation of the Code

The `save_design` function acts as the primary interface for generating the final design report. It integrates user inputs, design calculations, and report formatting into a cohesive workflow. The function's operation can be summarized in the following steps:

1. **Data Collection and Initialization:** The function begins by gathering all necessary design parameters from the user interface and internal class attributes. This includes material properties (e.g., f_u, f_y), geometric data (plate thickness, width), bolt specifications (grade, diameter, type), and applied loads (tensile or compressive forces).
2. **Design Status Validation:** Before proceeding, the function checks the `design_status` flag. If the design has failed (e.g., due to insufficient capacity or geometric constraints), it generates a report highlighting the failure mode. If the design is safe, it proceeds to compile the full successful design details.
3. **Latex Dictionary Construction:** A comprehensive dictionary, `report_input`, is constructed to map Python variables to their corresponding LaTeX display strings. This includes:

- **Connection Details:** Module name, loads, and material grades.
- **Design Checks:** Calculated capacities for shear (V_{dsb}), bearing (V_{dpb}), and tension (T_{db}).
- **Detailing Parameters:** Pitch (p), gauge (g), and edge distances (e).

4. **Dynamic Equation Formatting:** The function utilizes the `pylatex.utils.NoEscape` class to generate dynamic mathematical expressions. This allows the report to display the actual substitution of values into formulas (e.g., "$V_{dsb} = \frac{400 \times 245}{1.25} = 78.4$ kN") rather than just static text, providing transparency in the calculations.
5. **File Management and Compilation:** Finally, the function handles file system operations using the `os` module to create the required directories. It then invokes the `CreateLatex.save_latex` method to compile the inputs into a professional PDF document, complete with 3D images and proper formatting.

3.4 Task 2: Documentation

The directory structure used for the Report Generation is organized as follows:

```
src/osdag

 design_report/
    reportGenerator_latex.py

 design_type/
    connection/
       butt_joint_bolted.py

 gui
    ui_summary_popup.py/
    ui_template.py/
Report_functions.py
```

Program Start Calls The main entry point for the program is osdagMainPage.py. To start the program, open the Osdag folder and start the terminal with that path, execute the following command from the project root:

```
$ python -m osdag.osdagMainPage.py
```

3.5 References

- Osdag Github repository: https://osdag.github.io/
- IS 800:2007, "General Construction in Steel Code of Practice of Practice"

Chapter 4

Report Generation of Welded Lap Joint

4.1 Task 3: Problem Statement

The main objective was to generate a detailed design report for a **Welded Lap Joint Connection**. Using Osdag's modular structure, this tool automates the design checks, calculation steps, and capacity verifications for fillet-welded lap joints, strictly adhering to IS 800:2007.

4.2 Task 3: Tasks Done

4.2.1 Development of `save_design()` Function

* **Lead Function for Report Generation:** Constructed the `save_design()` function as the core orchestrator taking user input, performing verifications, and exporting data to LaTeX.
* **Dynamic Data Collection:** Parameters such as plate thicknesses, plate width, material grade, weld size, and load are retrieved from the interface and stored systematically.
* **Comprehensive Calculations:** The function runs mandatory design checks, including:

- **Weld Size Limit Check:** Ensures the weld lies within code-prescribed minimum and maximum limits, which depend on the thinner connected plate.
- **Weld Strength:**

$$f_{\text{wd}} = \frac{f_u \cdot a}{\sqrt{3}\gamma_{\text{mw}}}$$

where f_u is the weld/filler material strength, a is the effective throat thickness, and γ_{mw} is the weld partial safety factor.

- **Weld Length:** Required effective weld length calculated as

$$l_{\text{eff}} = \frac{\text{Applied Force}}{2f_{\text{wd}}}$$

- **Long Joint Reduction:** Reduces weld capacity for joints exceeding code-specified lengths.
- **Base Metal Strength:** Checks both gross section yielding and net section rupture as per IS 800 (Cl. 6.2, 6.3), including a shear lag factor for lap joints.
- **Utilization Ratio:**

$$UR = \frac{\text{Applied Load}}{\min(\text{Weld Capacity}, \text{Base Metal Capacity})}$$

* **Automated LaTeX Integration:** Results, equations, pass/fail results, detailing, and summary tables are mapped into a LaTeX template for PDF report generation using Osdag's `CreateLatex` system.

4.2.2 Check Implementation and Output

* **Logical Status Verification:** If any check fails (e.g., weld overstressed, length out of code bounds), the report highlights the cause with clear "Fail" status.
* **Summary Table Output:** The report summarizes all key checks (weld, base metal, utilization ratio) in a structured table for audit and review.
* **Deterministic and Reproducible:** The approach ensures consistency

of results and easy traceability, fulfilling code and project documentation requirements.

4.3 Task 3: Python Code

The entire logic for the report generation of Welded Lap Joint module is within the save_design() function. And it's being connected to the main GUI through ui_template.py and ui_popup_summary.py files. Below are key snippets from the module with explanations of their functionality.

Description of the Script

The script defines the save_design() function, which handles everything from user input to final design report generation. It uses reportGenerator_latex.py to manage properties and perform calculations.

Python Code

The following snippets showcase the core logic implemented in the report generation task.

Listing 4.1: Welded Lap Joint save_design() function

```
%--------------------begin code-------------
def save_design(self, *args):
        """
        Handles both old (save_design(main, popup_summary)) and
            new (save_design(popup_summary)) patterns.
        Generates compression/tension-aware report correctly.
        """
        # --- Handle arguments ---
        if len(args) == 1:
            popup_summary = args[0]
        elif len(args) == 2:
            _, popup_summary = args
        else:
            raise TypeError("save_design() expects 1 or 2
                arguments (main, popup_summary)")
```

```
14
15          # --- Report Input Parameters ---
16          force_label = "Compression Force (kN)" if self.
                design_for == 'Compression' else "Tensile Force (kN
                )"
17          self.report_input = {
18              "Module": self.module,
19              "Main Module": self.mainmodule,
20              force_label: round(self.axial_force / 1000, 2),
21              "Design Mode": self.design_for,
22              "Plate Thickness (mm)": f"{self.plate1.thickness
                    [0]}, {self.plate2.thickness[0]}",
23              "Material": self.main_material,
24              "Ultimate Strength, Fu (MPa)": self.plate1.fu,
25              "Yield Strength, Fy (MPa)": self.plate1.fy,
26              "Width of Plate (mm)": self.width,
27              "Weld Type": self.weld.type,
28              "Fabrication": self.weld.fabrication,
29              "Weld Size (mm)": self.weld_size,
30              "Effective Throat Thickness (mm)": getattr(self, '
                    effective_throat_thickness', None),
31              "End Return Length (mm)": getattr(self, '
                    end_return_length', None),
32              "Overlap Length (mm)": getattr(self, '
                    overlap_length', None),
33              "Gamma_mw": self.gamma_mw
34          }
35
36          # --- Report Checks Section ---
37          self.report_check = []
38          self.report_check.append(('SubSection', f'{self.
                design_for} Design Summary', '|p{3.2cm}|p{6.8cm}|p
                {5cm}|p{1cm}|'))
39
40          if self.design_for == 'Compression':
41              self.report_check.append((
42                  'Base Metal Capacity (Compression)',
43                  f"P_d = A_g    f_y /  _m0  = {self.A_g:.2f}
                        {self.plate1.fy}/{self.gamma_m0} = {self.
```

```
                    T_db/1000:.2f} kN [IS 800:2007, Cl.7.1.2]",
                f"{self.T_db/1000:.2f} kN",
                "Pass" if self.T_db >= self.axial_force else "
                    Fail"
            ))
        else:
            self.report_check.append((
                'Base Metal Capacity (Tension)',
                f"min(T_dg, T_dn) = min({self.A_g * self.plate1
                    .fy / self.gamma_m0 / 1000:.2f}, "
                f"{0.9 * self.A_g * self.plate1.fu * 0.7 / self
                    .gamma_m1 / 1000:.2f}) = {self.T_db/1000:.2
                    f} kN [IS 800:2007, Cl.6.2  6  .3]",
                f"{self.T_db/1000:.2f} kN",
                "Pass" if self.T_db >= self.axial_force else "
                    Fail"
            ))

        # Weld check
        self.report_check.append((
            f"Weld Strength ({self.design_for} Mode)",
            f"Required = {self.axial_force/1000:.2f} kN",
            f"Provided = {self.design_capacity/1000:.2f} kN",
            "Pass" if self.design_capacity >= self.axial_force
                else "Fail"
        ))

        # Utilization ratio
        self.report_check.append((
            'Overall Utilization Ratio',
            "<= 1.0",
            f"{self.utilization_ratio:.3f}",
            "Pass" if self.utilization_ratio <= 1.0 else "Fail"
        ))

        # --- Generate PDF ---
        fname_no_ext = popup_summary['filename']
        CreateLatex.save_latex(
            CreateLatex(),
```

```
76              self.report_input,
77              self.report_check,
78              popup_summary,
79              fname_no_ext,
80              os.path.abspath(".").replace("\\", "/"),
81              [],
82              "/ResourceFiles/images/3d.png",
83              module=self.module
84          )
85          return True
86  %-------------------- end code ---------------
```

4.3.1 Explanation of the Code

The `save_design` function for the welded lap joint is the central routine that converts the completed design into a structured LaTeX report. Its workflow can be summarized as follows:

1. **Collect Design Data:** All relevant input and output values stored in the welded lap joint object are gathered. This includes: plate dimensions (thickness and width), weld size, weld length, material properties (f_y, f_u), applied load, and intermediate results such as weld capacity, base metal capacity, and utilization ratio.
2. **Check Design Status:** The function reads the internal `design_status` flag:
   - If the design has failed (for example, required weld length exceeds available overlap, or $UR > 1.0$), it prepares a brief failure report explaining the governing reason.
   - If the design is safe, it proceeds to create a full, detailed report.
3. **Prepare `report_input` and Check List:** A dictionary is assembled to pass all data to the LaTeX template. Typical items are:
   - Connection description and input summary,
   - Stepwise weld design: permissible weld size range, design weld strength f_{wd}, required and provided weld length, long-weld reduction (if any),

- Base metal strength checks (gross yielding, net rupture),
- Final utilization ratio and pass/fail remarks.

Along with this, a list of design checks (DDCL-style) is created so that each check can be printed with its status in the report.

4. **Generate LaTeX Equations:** Using LaTeX strings (often wrapped with `NoEscape`), the function formats the main formulas with substituted numerical values, such as:

$$f_{\mathrm{wd}} = \frac{f_u \cdot a}{\sqrt{3}\,\gamma_{\mathrm{mw}}}, \qquad R_{\mathrm{weld}} = 2\,l_{\mathrm{eff}}\,f_{\mathrm{wd}},$$

and the comparison of applied load with weld and plate capacities. This makes the report transparent and easy to follow.

5. **File Handling and Report Creation:** Finally, the function sets up the output folder (creating it if needed) and calls the common `CreateLatex.save_latex` method. This compiles the LaTeX document into a professional PDF that contains: input summary, calculation steps, DDCL table, and conclusion for the welded lap joint.

4.4 Task 3: Documentation

The directory structure used for the Report Generation is organized as follows:

```
src/osdag

 design_report/
    reportGenerator_latex.py

 design_type/
    connection/
       lap_joint_welded.py

 gui
    ui_summary_popup.py/
```

ui_template.py/

Report_functions.py

Program Start Calls The main entry point for the program is osdagMainPage.py. To start the program, open the Osdag folder and start the terminal with that path, execute the following command from the project root:

```
$ python -m osdag.osdagMainPage.py
```

4.5 References

* Osdag Github repository: https://osdag.github.io/
* IS 800:2007, "General Construction in Steel Code of Practice of Practice"

Chapter 5

Report Generation of Welded Butt Joint

5.1 Task 4: Problem Statement

The objective of this task was to develop an automated design and reporting workflow for a **Welded Butt Joint** connection in Osdag. The module must design the weld and base metal in accordance with IS 800:2007, verify all limit states (weld strength, long joint reduction, base metal yielding and rupture) and generate a clear, stepwise LaTeX report for documentation and checking purposes.

5.2 Task 4: Tasks Done

5.2.1 Design Automation Workflow

- Implemented the `ButtJointWelded` class as a moment connection module, handling all required inputs: plate thicknesses, plate width, material grade, weld size, cover plate option, and axial force.
- Added UI input, detailing and weld preference functions (e.g. `input_values`, `weld_values`, `detailing_values`) to collect user choices such as shop/-field weld and weld material strength.
- Implemented validation in `func_for_validation` to ensure non-empty

and positive values for key fields (plate width, axial force) before starting the design.

5.2.2 Core Design Functions

- **Initialisation (`set_input_values`):** Sets up plates, weld object, design force (tension or compression), cover plate type (single/double) and calculates required cover plate thickness and packing plate thickness as per IS 800:2007 recommendations.
- **Design of Weld (`design_of_weld`):**
- Selects a valid weld size within IS 800 limits using the minimum and maximum weld size rules.
- Computes effective throat thickness and design weld strength

$$f_w = \frac{f_u}{\sqrt{3}\,\gamma_{mw}}.$$

- Calls subsequent functions for weld length, strength verification, long joint reduction and base metal checks.
- **Weld Length (`weld_length`):**
- Checks that the chosen weld size is between the computed $s_{\min}$ and $s_{\max}$.
- Determines required weld length from the design axial force; if possible a straight weld equal to plate width is used, otherwise skewed welds and possible side welds are calculated.
- Stores the provided and effective weld length and weld angle for later reporting.
- **Weld Strength Verification (`weld_strength_verification`):**
- Ensures effective weld length satisfies the minimum requirement $L_{\text{eff}} \geq 4s$.
- Computes weld strength

$$P_w = N_f \cdot 0.707\, s \cdot L_{\text{eff}} \cdot f_w$$

and the weld utilization ratio $UR_{\text{weld}} = P_{\text{req}}/P_w$.

- **Long Joint Reduction (`long_joint_reduction_factor`):**
- For long welds, calculates reduction factor β_L based on L_{eff} and effective throat thickness, and updates the reduced weld strength and utilization.
- **Base Metal Strength (`check_base_metal_strength`):**
- Computes gross area and net area (equal for welded joints) and evaluates:

$$T_{dy} = \frac{A_g f_y}{\gamma_{m0}}, \quad T_{du} = \frac{0.9 A_n f_u}{\gamma_{m1}}$$

  for tension, or compression capacity for compression design.
- Stores base metal utilization ratio.
- **Final Check (`calculate_final_utilization_ratio`):**
- Takes the maximum of weld and base metal utilization ratios as the overall utilization.
- Sets `design_status` to SAFE if all utilization ratios are ≤ 1.0, otherwise UNSAFE with guidance on which component governs.

5.2.3 Report Generation with `save_design`

- The `save_design` function is called only when `design_status` is SAFE. It gathers all important data (forces, material strengths, plate dimensions, weld size, cover plate type) into a `report_input` dictionary.
- It then constructs a detailed `report_check` list containing:
- Weld size limits ($s_{\min}$, $s_{\max}$) and pass/fail status,
- Weld strength calculation with substituted values,
- Base metal capacity in tension or compression,
- Overall capacity, utilization ratio and final design status.
- Mathematical expressions are wrapped using `NoEscape` so that LaTeX formulas (for example the expression of f_w or P_w) appear correctly in the final PDF.

- Finally, `CreateLatex.save_latex` is invoked to generate the formatted PDF report, including 2D/3D images and a summary table of all checks.

5.3 Task 4: Python Code

The entire logic for the report generation of Welded Butt Joint module is within the save_design() function. And it's being connected to the main GUI through ui_template.py and ui_popup_summary.py files. Below are key snippets from the module with explanations of their functionality.

Description of the Script

The script defines the save_design() function, which handles everything from user input to final design report generation. It uses reportGenerator_latex.py to manage properties and perform calculations.

Python Code

The following snippets showcase the core logic implemented in the report generation task.

Listing 5.1: Welded Butt Joint save_design() function

```
1  %--------------------begin code-------------
2  def save_design(self, popup_summary):
3          """Enhanced save_design with proper compression/
               tension handling and LaTeX formatting"""
4          import os as os_module
5          import math
6          from pylatex.utils import NoEscape
7
8          try:
9              if not self.design_status:
10                 print("ERROR: Cannot generate report -
                       design is not complete or failed")
11                 return False
12
```

```
13              def safe_get(attr, default=0):
14                  value = getattr(self, attr, default)
15                  return default if value is None else value
16
17              # Basic info extraction
18              design_for = safe_get('design_for', 'Tension')
19              is_compression = design_for.lower().startswith(
                    'c')
20              force = safe_get('axial_force', safe_get('
                    tensile_force', 0))
21              force_kN = round(force / 1000, 2)
22
23              plate1_thk = safe_get('plate1.thickness[0]',
                    8.0)
24              plate2_thk = safe_get('plate2.thickness[0]',
                    8.0)
25              t_min = min(plate1_thk, plate2_thk)
26              fy = safe_get('fy', 250)
27              fu = safe_get('fu', 410)
28              gamma_m0 = safe_get('gamma_m0', 1.1)
29              gamma_m1 = safe_get('gamma_m1', 1.25)
30              gamma_mw = safe_get('gamma_mw', 1.25)
31              weld_size = safe_get('weld_size', 6.0)
32              plates_width = safe_get('plates_width', 40.0)
33              cover_plate = safe_get('cover_plate', 'Single-
                    Cover')
34
35              # Prepare header data
36              self.report_input = {
37                  KEY_MODULE: getattr(self, 'module', 'Butt
                        Joint Welded Connection'),
38                  KEY_DISP_MATERIAL: safe_get('main_material'
                        , 'E 250 (Fe 410 W)A'),
39                  KEY_DISP_AXIAL: force_kN,
40                  KEY_DISP_DESIGN_FOR: design_for,
41                  KEY_DISP_PLATE1_THICKNESS: plate1_thk,
42                  KEY_DISP_PLATE2_THICKNESS: plate2_thk,
43                  KEY_DISP_PLATE_WIDTH: plates_width,
44                  KEY_DISP_COVER_PLT: cover_plate,
```

```
                KEY_DISP_WELD_SIZE: weld_size,
                KEY_DISP_DP_WELD_TYPE: safe_get('weld_type'
                    , 'Shop weld'),
                KEY_DISP_DP_WELD_MATERIAL_G_O_REPORT: fu,
                KEY_DISP_ULTIMATE_STRENGTH_REPORT: fu,
                KEY_DISP_YIELD_STRENGTH_REPORT: fy,
                KEY_DISP_GAMMA_M0: gamma_m0,
                KEY_DISP_GAMMA_M1: gamma_m1,
                KEY_DISP_GAMMA_MW: gamma_mw
            }

            self.report_check = []

            # ------------------------------------
            # Section: Weld Design
            # ------------------------------------
            t1 = ('SubSection', 'Weld Design', '|p{4cm}|p{6
                cm}|p{4cm}|p{2cm}|')
            self.report_check.append(t1)

            s_min = safe_get('s_min', 3)
            s_max = safe_get('s_max', 6.5)
            f_w = fu / (math.sqrt(3) * gamma_mw)
            te = 0.707 * weld_size
            N_f = 2 if "double" in cover_plate.lower() else
                1
            L_req = force / (N_f * te * f_w) if (N_f * te *
                f_w) > 0 else 0

            self.report_check.append((
                'Minimum Weld Size',
                NoEscape(f'\\small s$_{{min}}$ = {s_min} mm
                    [Ref: IS 800:2007, Table 21]'),
                f'{weld_size} mm',
                'Pass' if weld_size >= s_min else 'Fail'
            ))

            self.report_check.append((
                'Maximum Weld Size',
```

```
                NoEscape(f'\\small s$_{{max}}$ = {s_max} mm [Ref: IS 800:2007, Cl.10.5.3.1]'),
                f'{weld_size} mm',
                'Pass' if weld_size <= s_max else 'Fail'
            ))

            self.report_check.append((
                'Weld Strength',
                NoEscape(f'\\small f$_w$ = f$_u$ / ( 3 $_ {{mw}}$) = {fu}/( 3 {gamma_mw}) = {round(f_w,2)} N/mm '),
                f'{round(f_w,2)} N/mm ',
                'Pass'
            ))

            L_eff_min = 4 * weld_size
            L_eff_provided = plates_width - (2 * weld_size)
            self.report_check.append((
                'Effective Weld Length',
                NoEscape(f'\\small L$_eff$ 4s = 4 {weld_size} = {L_eff_min} mm'),
                f'{round(L_eff_provided, 2)} mm',
                'Pass' if L_eff_provided >= L_eff_min else 'Fail'
            ))

            weld_strength = N_f * te * L_eff_provided * f_w
            self.report_check.append((
                'Weld Strength Verification',
                NoEscape(f'\\small P$_w$ = N$_f$ t$_e$ L$_eff$ f$_w$ = {N_f} {round(te,2)} {round(L_eff_provided,2)} {round(f_w,2)} = {round(weld_strength/1000,2)} kN'),
                f'{round(weld_strength/1000,2)} kN',
                'Pass' if weld_strength >= force else 'Fail'
            ))

```

```
108             # ------------------------------------
109             # Section: Base Metal Strength
110             # ------------------------------------
111             self.report_check.append(('SubSection', 'Base
                    Metal Strength', '|p{4cm}|p{6cm}|p{4cm}|p{2
                    cm}|'))
112
113             A_g = t_min * plates_width
114             if is_compression:
115                 # Compression check as per Cl. 7.1.2
116                 P_d = A_g * fy / gamma_m0
117                 self.report_check.append((
118                     'Base Metal Capacity (Compression)',
119                     NoEscape(f'\\small P$_d$ = A$_g$
                        f$_y$/ $_ {{m0}}$ = {A_g:.1f}  {fy
                        }/{gamma_m0} = {round(P_d/1000,2)}
                        kN [IS 800:2007, Cl.7.1.2]'),
120                     f'{round(P_d/1000,2)} kN',
121                     'Pass' if P_d >= force else 'Fail'
122                 ))
123                 T_db = P_d
124             else:
125                 # Tension as per Cl. 6.2, 6.3
126                 T_dy = A_g * fy / gamma_m0
127                 T_du = 0.9 * A_g * fu / gamma_m1
128                 T_db = min(T_dy, T_du)
129                 self.report_check.append((
130                     'Base Metal Capacity (Tension)',
131                     NoEscape(f'\\small min(T$_{{dy}}$, T$_
                        {{du}}$) = min({round(T_dy/1000,2)},
                         {round(T_du/1000,2)}) = {round(T_db
                        /1000,2)} kN [IS 800:2007, Cl.6.2
                          6  .3]'),
132                     f'{round(T_db/1000,2)} kN',
133                     'Pass' if T_db >= force else 'Fail'
134                 ))
135
136             # ------------------------------------
137             # Section: Summary
```

```
138             # ------------------------------------
139             self.report_check.append(('SubSection', 'Design
                    Summary', '|p{4cm}|p{6cm}|p{4cm}|p{2cm}|'))
140
141             overall_capacity = min(weld_strength, T_db)
142             utilization = force / overall_capacity if
                   overall_capacity > 0 else 0
143
144             self.report_check.append((
145                 'Overall Capacity',
146                 NoEscape(f'\\small min(Weld, Base Metal) =
                        min({round(weld_strength/1000,2)}, {
                        round(T_db/1000,2)}) = {round(
                        overall_capacity/1000,2)} kN'),
147                 f'{round(overall_capacity/1000,2)} kN',
148                 'Pass'
149             ))
150             self.report_check.append((
151                 'Utilization Ratio',
152                 NoEscape(f'\\small    = P / C = {round(
                        force_kN,2)} / {round(overall_capacity
                        /1000,2)} = {round(utilization,3)}'),
153                 f'{round(utilization,3)}',
154                 'Pass' if utilization <= 1.0 else 'Fail'
155             ))
156
157             self.report_check.append((
158                 'Overall Design Status',
159                 'All design checks must pass',
160                 'SAFE' if utilization <= 1.0 else 'UNSAFE',
161                 'Pass' if utilization <= 1.0 else 'Fail'
162             ))
163
164             # ------------------------------------
165             # Report metadata and PDF creation
166             # ------------------------------------
167             popup_summary.setdefault('ProjectTitle', 'Butt
                   Joint Welded Connection Report')
168             popup_summary.setdefault('Subtitle', '
```

```
                Structural Steel Connection Design')
169             popup_summary.setdefault('Client', 'Client Name
                ')
170             popup_summary.setdefault('AdditionalComments',
                'All design checks conform to IS 800:2007
                provisions.')
171
172             fname_no_ext = popup_summary.get('filename', '
                Butt_Joint_Welded_Report')
173             folder = popup_summary.get('folder', './reports
                ')
174             os_module.makedirs(folder, exist_ok=True)
175
176             from ...design_report.reportGenerator_latex
                import CreateLatex
177             latex = CreateLatex()
178             Disp_2d_image = [
179                 "/ResourceFiles/images/top.png",
180                 "/ResourceFiles/images/side.png",
181                 "/ResourceFiles/images/front.png"
182             ]
183
184             Disp_3D_image = "/ResourceFiles/images/3d.png"
185
186             latex.save_latex(
187                 self.report_input,
188                 self.report_check,
189                 popup_summary,
190                 fname_no_ext,
191                 os.path.abspath(".").replace("\\", "/"),
192                 Disp_2d_image,
193                 Disp_3D_image,
194                 getattr(self, 'module', 'ButtJointWelded')
195             )
196
197             pdf_path = os_module.path.join(folder, f"{
                fname_no_ext}.pdf")
198             if os_module.path.exists(pdf_path):
199                 print(f"SUCCESS: Report generated at {
```

```
                    pdf_path}")
200                 return True
201             else:
202                 print("ERROR: Report PDF not found")
203                 return False
204
205         except Exception as e:
206             print(f"CRITICAL ERROR in save_design(): {e}")
207             return False
208         return True
209 %-------------------- end code ---------------
```

5.3.1 Explanation of the Code

The `save_design` function for the welded butt joint is responsible for converting the completed design into a structured LaTeX report. Its main steps are:

1. **Collecting Input and Output Data:** All relevant information stored in the `ButtJointWelded` object is gathered. This includes: plate dimensions and material properties, weld size and type, cover plate details (if any), design axial force, weld and base metal capacities, and final utilization ratios.
2. **Checking Design Status:** The function reads the internal `design_status` flag.
   - If the design is *UNSAFE*, it prepares a short report summarizing the failure reason (for example, "weld capacity required factored load" or "base metal utilization > 1.0").
   - If the design is *SAFE*, it proceeds to create a complete, stepwise report.
3. **Preparing `report_input` and Design Checks:** A dictionary `report_input` is created to pass all numerical values and text labels to the LaTeX template. Typical entries cover:
   - Connection description and input summary,

- Weld design: permissible weld size range, chosen weld size, design weld strength, required and provided weld length, long-joint reduction factor (if applicable),
- Base metal strength in tension or compression,
- Final design strength and overall utilization ratio.

In parallel, a list of design checks (DDCL-style) is assembled, where each check stores its name, governing formula, numerical substitution and PASS/FAIL status for tabular presentation.

4. **Formatting LaTeX Equations:** Important formulas are written as LaTeX math strings, often wrapped with `NoEscape`, for example:

$$f_w = \frac{f_u}{\sqrt{3}\,\gamma_{mw}}, \qquad P_{\text{weld}} = A_{\text{weld}}\, f_w,$$

and the comparison of applied force with weld and plate capacities. This allows the report to display both the general equation and the corresponding numerical substitution.

5. **File Handling and PDF Generation:** Finally, the function sets up the output directory (creating it if required), and calls the common `CreateLatex.save_latex` routine. This compiles the LaTeX file into a professional PDF report that includes input summary, detailed calculation steps, design check table and conclusion for the welded butt joint.

5.4 Task 4: Documentation

The directory structure used for the Report Generation is organized as follows:

```
src/osdag

 design_report/
    reportGenerator_latex.py
```

```
design_type/
    connection/
        butt_joint_welded.py


gui
    ui_summary_popup.py/
    ui_template.py/
Report_functions.py
```

Program Start Calls The main entry point for the program is `osdagMainPage.py`. To start the program, open the Osdag folder and start the terminal with that path, execute the following command from the project root:

```
$ python -m osdag.osdagMainPage.py
```

5.5 References

- Osdag Github repository: https://osdag.github.io/
- IS 800:2007, "General Construction in Steel Code of Practice of Practice"

Chapter 6

Conclusions

6.1 Tasks Accomplished

During the course of the screening task and report generation , the following key tasks were accomplished:

- Developed the design report generation mechanism for the **Bolted Lap Joint** Connections.
- Developed the design report generation mechanism for the **Bolted Butt Joint** Connections.
- Developed the design report generation mechanism for the **Welded Lap Joint** Connections.
- Updated the design report generation of the **Welded Butt Joint** Connections by adding both compression and tensile forces.

6.2 Skills Developed

Working on the bolted lap joint, bolted butt joint, welded lap joint and welded butt joint modules helped in developing a broad set of technical and professional skills. These can be grouped as follows:

1. Advanced Python Programming and Software Engineering

- **Modular and Object-Oriented Design:** Designed separate classes and functions for each connection type, clearly separating responsibilities such as input handling, design calculations, checks, and report generation. This improved understanding of encapsulation, cohesion, and reusability in engineering software.
- **Clean Data Flow and Validation:** Implemented structured methods to collect and validate inputs (plate dimensions, bolt/weld properties, loads). This required designing logical validation rules, managing default values, and giving clear feedback on invalid data, which is essential for robust engineering tools.
- **Error Handling and Robustness:** Used conditional checks and try–except patterns to handle possible runtime issues (missing folders, invalid numeric values, division by zero, failed designs), improving the resilience of the code and preventing abrupt program termination.
- **Use of Python Standard Libraries:** Gained practical experience with modules such as `os` and `math` for file-path handling, directory creation, and numerical operations, which are frequently needed in real engineering applications.

2. LaTeX-Based Report Generation and Mathematical Communication

- **Automated LaTeX Report Creation:** Learned how to convert raw numerical results into a structured engineering report by building dictionaries and lists that feed directly into a LaTeX template through the Osdag reporting framework.
- **Dynamic Equation Formatting:** Used tools like `pylatex` and `NoEscape` to embed Python variables into LaTeX math expressions. This made it possible to show both general design formulas and their

numerical substitutions, improving transparency of calculations.

· **Clear Presentation of Design Checks:** Structured the output so that each important check (bolt shear, bearing, weld strength, base-metal yielding, block shear, long-joint reduction, detailing limits) appears with its equation, substituted values, and PASS/FAIL status. This strengthened skills in technical documentation and communication.

· **Professional Engineering Documentation:** Gained practice in organising chapters, sections, tables and equations in a consistent format, which is essential for preparing professional design reports, theses and submission documents.

3. Structural Design and Code Interpretation Skills

· **Understanding of IS 800:2007 Provisions:** Translated clauses related to tension members, bolted and welded connections into algorithms. This deepened understanding of concepts such as gross section yielding, net section rupture, block shear, weld design strength, and capacity reduction factors.

· **Connection Design Logic:** Implemented stepwise design procedures for:

· Bolted connections: shear capacity, bearing capacity, packing plate and long-joint effects, and detailing limits (pitch, gauge, edge distance).

· Welded connections: weld size limits, effective throat, effective length, long joint reduction and base-metal checks.

This strengthened the ability to move from codal equations to executable design logic.

· **Utilization Ratio and Safety Assessment:** Developed a consistent approach for calculating utilization ratios and identifying the governing failure mode. This improved the ability to judge the adequacy of a design and to clearly communicate safety margins.

- **Practical Detailing Awareness:** By coding checks for minimum/-maximum edge distances, pitch, gauge, weld length and weld size, gained an appreciation of how theoretical design interacts with fabrication and erection constraints in real projects.

4. General Professional and Problem-Solving Skills

- **Algorithmic Thinking:** Broke down complex design procedures into smaller logical steps and implemented them systematically, enhancing problem-solving and structured thinking.
- **Debugging and Verification:** Compared intermediate results with hand calculations and corrected inconsistencies. This improved attention to detail and the ability to verify and validate engineering software.
- **Interdisciplinary Integration:** Combined knowledge of structural engineering, programming and technical writing to deliver a complete tool chain from input to final report, reflecting the type of integrated work expected in professional practice.

Chapter A

Appendix

A.1 Work Reports

Internship Work Report			
NAME	Roushan Raj		
PROJECT	Osdag		
INTERNSHIP	FOSSEE Semester Long Fellowship 2025		
DATE	**DAY**	**TASK**	**HOURS WORKED**
08/01/25	Friday	Initial analysis, requirements gathering and module planning with mentor.	4
08/02/25	Saturday	Set up class structure, inheritance from MomentConnection and basic UI framework.	5
08/03/25	Sunday	Holiday	0
08/04/25	Monday	Implemented input_values() and customized_input() methods for bolt and plate parameters.	8
08/05/25	Tuesday	Developed bolt selection logic (select_bolt_dia_and_grade) with capacity calculations.	5
08/06/25	Wednesday	Coded bolt arrangement algorithm (number_r_c_bolts) with iterative column/row calculation.	6
08/07/25	Thursday	Implemented capacity reduction checks (long joint and large grip factors).	4
08/08/25	Friday	Implemented basic save_design skeleton for Bolted Lap Joint module.	5
08/09/25	Saturday	Developed base metal strength checks (tension and compression modes).	5
08/10/25	Sunday	Holiday	0
08/11/25	Monday	Integrated utilization ratio calculation and final design status logic.	6
08/12/25	Tuesday	Unit testing of bolt capacity and plate strength calculations.	4
08/13/25	Wednesday	Integration testing of complete design workflow with sample inputs.	5
08/14/25	Thursday	Implemented save_design() function for LaTeX report generation.	6
08/15/25	Friday	Formatted mathematical equations and substitution values for LaTeX output.	7
08/16/25	Saturday	Created design verification checklist and refined error handling logic.	6
08/17/25	Sunday	Holiday	0
08/18/25	Monday	Code review, bug fixes and documentation of Bolted Lap Joint module.	4
08/19/25	Tuesday	Analysis of bolted butt joint design requirements and cover plate logic.	5
08/20/25	Wednesday	Designed class structure and input methods for bolted butt joint module.	6
08/21/25	Thursday	Implemented cover plate thickness calculations (single and double cover).	4

08/22/25	Friday	Coded packing plate logic and beta_pkg reduction factor implementation.	7
08/23/25	Saturday	Developed bolt capacity calculations adapted for butt joint configuration.	8
08/24/25	Sunday	Holiday	0
08/25/25	Monday	Implemented spacing checks and detailing requirements (pitch, gauge, edge distance).	5
08/26/25	Tuesday	Coded base metal strength checks for tension and compression in butt configuration.	7
08/27/25	Wednesday	Integrated capacity reduction factors (long joint and large grip effects).	6
08/28/25	Thursday	Implemented utilization ratio tracking for both bolts and base metal.	5
08/29/25	Friday	Unit testing of cover plate calculations and bolt arrangement logic.	4
08/30/25	Saturday	Integration testing and validation against manual calculations.	6
08/31/25	Sunday	Holiday	0
09/01/25	Monday	Implemented save_design() function with LaTeX formatting for butt joint.	5
09/02/25	Tuesday	Testing of report generation and PDF output with sample design cases.	5
09/03/25	Wednesday	Code review, debugging and final refinements for Bolted Butt Joint module.	5
09/04/25	Thursday	Requirements analysis and design approach planning for welded lap joint.	5
09/05/25	Friday	Set up class structure and weld preference methods (weld_values, detailing_values).	5
09/06/25	Saturday	Implemented weld size validation using IS 800: 2007 Table 21 limits.	6
09/07/25	Sunday	Holiday	0
09/08/25	Monday	Developed weld strength calculations (fillet weld design strength).	4
09/09/25	Tuesday	Coded effective throat thickness and design weld strength formulas.	4
09/10/25	Wednesday	Implemented weld length calculations (minimum and maximum effective lengths).	4
09/11/25	Thursday	Developed long joint reduction factor calculation (beta_lw).	4
09/12/25	Friday	Implemented base metal strength checks for tension (yielding and rupture).	6
09/13/25	Saturday	Coded utilization ratio calculations and final design status logic.	6
09/14/25	Sunday	Holiday	0
09/15/25	Monday	Unit testing of weld design functions and strength calculations.	7
09/16/25	Tuesday	Integration testing of complete welded lap joint design workflow.	5
09/17/25	Wednesday	Implemented save_design() function with detailed LaTeX formatting.	6

09/18/25	Thursday	Formatted weld equations and substitution values for professional report output.	2
09/19/25	Friday	Testing of report generation with various load and weld size combinations.	7
09/20/25	Saturday	Code review, documentation and debugging of Welded Lap Joint module.	5
09/21/25	Sunday	Holiday	0
09/22/25	Monday	Final refinements and quality assurance for welded lap joint module.	3
09/23/25	Tuesday	Requirements analysis and design specifications for welded butt joint.	5
09/24/25	Wednesday	Class structure design and UI methods implementation for butt joint.	4
09/25/25	Thursday	Coded weld size validation and effective throat thickness calculations.	6
09/26/25	Friday	Implemented weld length determination (straight, skewed and side welds).	5
09/27/25	Saturday	Developed weld strength verification with minimum length checks.	4
09/28/25	Sunday	Holiday	0
09/29/25	Monday	Created report_input structure for Welded Butt Joint save_design.	8
09/30/25	Tuesday	Implemented long joint reduction factor for butt weld configuration.	8
10/01/25	Wednesday	Coded base metal strength checks (tension and compression modes).	7
10/02/25	Thursday	Implemented utilization ratio tracking for weld and base metal components.	6
10/03/25	Friday	Unit testing of weld calculations and base metal strength functions.	5
10/04/25	Saturday	Integration testing with various design scenarios and load combinations.	4
10/05/25	Sunday	Holiday	0
10/06/25	Monday	Implemented save_design() function with comprehensive LaTeX reporting.	5
10/07/25	Tuesday	Formatted equations and output tables for professional report generation.	6
10/08/25	Wednesday	Testing of report generation and PDF compilation with multiple design cases.	7
10/09/25	Thursday	Code review and debugging of Welded Butt Joint module functions.	5
10/10/25	Friday	Final refinements and quality assurance for welded butt joint module.	4
10/11/25	Saturday	Comprehensive integration testing across all four connection modules.	8
10/12/25	Sunday	Holiday	0
10/13/25	Monday	Cross-module validation and consistency checks for design results.	5
10/14/25	Tuesday	Performance testing and optimization of module execution times.	5

10/15/25	Wednesday	Started writing Chapter 4 for internship report (Bolted Lap Joint design).	6
10/16/25	Thursday	Documented design approach and calculations for Bolted Lap Joint chapter.	6
10/17/25	Friday	Wrote problem statement and tasks completed sections for Bolted Lap Joint.	7
10/18/25	Saturday	Created LaTeX equations and formatted technical content for chapter 4.	4
10/19/25	Sunday	Holiday	0
10/20/25	Monday	Started Chapter 4 content for Bolted Butt Joint module documentation.	4
10/21/25	Tuesday	Wrote problem statement for Bolted Butt Joint chapter with design flow.	6
10/22/25	Wednesday	Documented capacity reduction factors and detailing requirements.	6
10/23/25	Thursday	Started documentation for Welded Lap Joint chapter with design principles.	6
10/24/25	Friday	Documented weld design calculations and strength verification procedures.	6
10/25/25	Saturday	Wrote problem statement and methodology for welded lap joint chapter.	6
10/26/25	Sunday	Holiday	0
10/27/25	Monday	Documented long joint reduction and base metal checks for lap weld.	5
10/28/25	Tuesday	Started Welded Butt Joint chapter documentation with design approach.	5
10/29/25	Wednesday	Documented skewed weld logic and cover plate integration in butt joint.	5
10/30/25	Thursday	Wrote design verification and utilization ratio sections for butt weld.	6
10/31/25	Friday	Created comprehensive save_design() function explanations in LaTeX.	6
11/01/25	Saturday	Documented skills developed section with technical achievements.	4
11/02/25	Sunday	Holiday	0
11/03/25	Monday	Formatted all chapters with consistent LaTeX structure and citations.	5
11/04/25	Tuesday	Compiled all four module chapters into unified report structure.	6
11/05/25	Wednesday	Created table of contents and cross-references between chapters.	7
11/06/25	Thursday	Added introduction and background sections for steel connection design.	4
11/07/25	Friday	Implemented consistent formatting and mathematical typesetting throughout.	8
11/08/25	Saturday	Proof-reading and technical accuracy verification of all chapters.	2
11/09/25	Sunday	Holiday	0
11/10/25	Monday	Final editing and refinement of report content and structure.	2

11/11/25	Tuesday	Final comprehensive testing of all four modules with edge cases.	3
11/12/25	Wednesday	Validation against hand-calculated design examples and IS 800:2007 standards.	4
11/13/25	Thursday	Testing error handling and user input validation across all modules.	5
11/14/25	Friday	Performance optimization and code efficiency improvements.	6
11/15/25	Saturday	Final bug fixes and refinements based on testing results.	7
11/16/25	Sunday	Holiday	0
11/17/25	Monday	Final proofreading and formatting of internship report.	5
11/18/25	Tuesday	Prepared executive summary and conclusions for final submission.	6
11/19/25	Wednesday	Generated PDF outputs and verified all links and cross-references.	7
11/20/25	Thursday	Final review with mentor and incorporation of feedback.	4
11/21/25	Friday	Made final edits and prepared all deliverable files for submission.	2
11/22/25	Saturday	Completed final documentation.	3
11/23/25	Sunday	Holiday	0
11/24/25	Monday	Committed all the changes to the specific remote branch.	5
11/25/25	Tuesday	Worked on shifting the code to another branch as directed by the mentor.	2
11/26/25	Wednesday	Rebased my local repo with other's remote and fixed merge conflicts.	3
11/27/25	Thursday	Made the final commits to the designated branch.	4
11/28/25	Friday	Final task review and submission	2
11/29/25	Saturday	Fellowship conclusion and final wrap-up	1
11/30/25	Sunday	Holiday	0

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.