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

July 31, 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.
- I wish to express my sincere appreciation to my colleagues, **Srinivas Raghav V C and Sushree Sangita**. Their commitment and collaborative spirit significantly eased the challenges of report generation and made this fellowship a both fruitful and enjoyable experience.

- 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 **5**

1.1 National Mission in Education through ICT 5

1.1.1 ICT Initiatives of MoE 6

1.2 FOSSEE Project 7

1.2.1 Projects and Activities 7

1.2.2 Fellowships 7

1.3 Osdag Software 8

1.3.1 Osdag GUI 9

1.3.2 Features 9

2 Screening Task **10**

2.1 Problem Statement 10

2.2 Tasks Done 10

2.3 Python Code 12

3 Reimagining Osdag Interface **26**

3.1 Task 1: Problem Statement 26

3.2 Task 1: Tasks Done 26

3.3 Thinking philosophy 27

4 Report Generation of Welded Butt Joint **30**

4.1 Task 1: Problem Statement 30

4.2 Task 1: Tasks Done 30

4.2.1 Core Save Design Function Development 30

4.2.2 Detailed Design Verification and Check Implementation 32

4.2.3 Advanced LaTeX Mathematical Formatting 33

4.2.4 Design Calculations Integration 34

4.2.5 UI and Integration Tasks 35

4.3 Task 1: Python Code 35

4.3.1 Explanation of the Code 1 49

4.3.2 Explanation of the Code 2 54

4.3.3 Full code integration 54
4.3.4 Explanation of the Code 61
4.4 Task 1: Documentation 61
4.5 References 62

5 Conclusions **63**
5.1 Tasks Accomplished 63
5.2 Skills Developed 63

A Appendix **65**
A.1 Work Reports 65

Bibliography **69**

Chapter 1

Introduction

1.1 National Mission in Education through ICT

The National Mission on Education through ICT (NMEICT) is a scheme under the Department of Higher Education, Ministry of Education, Government of India. It aims to leverage the potential of ICT to enhance teaching and learning in Higher Education Institutions in an anytime-anywhere mode.

The mission aligns with the three cardinal principles of the Education Policy—**access, equity, and quality**—by:

- Providing connectivity and affordable access devices for learners and institutions.
- Generating high-quality e-content free of cost.

NMEICT seeks to bridge the digital divide by empowering learners and teachers in urban and rural areas, fostering inclusivity in the knowledge economy. Key focus areas include:

- Development of e-learning pedagogies and virtual laboratories.
- Online testing, certification, and mentorship through accessible platforms like EduSAT and DTH.
- Training and empowering teachers to adopt ICT-based teaching methods.

For further details, visit the official website: www.nmeict.ac.in.

1.1.1 ICT Initiatives of MoE

The Ministry of Education (MoE) has launched several ICT initiatives aimed at students, researchers, and institutions. The table below summarizes the key details:

No.	Resource	For Students/Researchers	For Institutions
Audio-Video e-content			
1	SWAYAM	Earn credit via online courses	Develop and host courses; accept credits
2	SWAYAMPRABHA	Access 24x7 TV programs	Enable SWAYAMPRABHA viewing facilities
Digital Content Access			
3	National Digital Library	Access e-content in multiple disciplines	List e-content; form NDL Clubs
4	e-PG Pathshala	Access free books and e-content	Host e-books
5	Shodhganga	Access Indian research theses	List institutional theses
6	e-ShodhSindhu	Access full-text e-resources	Access e-resources for institutions
Hands-on Learning			
7	e-Yantra	Hands-on embedded systems training	Create e-Yantra labs with IIT Bombay
8	FOSSEE	Volunteer for open-source software	Run labs with open-source software
9	Spoken Tutorial	Learn IT skills via tutorials	Provide self-learning IT content
10	Virtual Labs	Perform online experiments	Develop curriculum-based experiments
E-Governance			
11	SAMARTH ERP	Manage student lifecycle digitally	Enable institutional e-governance
Tracking and Research Tools			
12	VIDWAN	Register and access experts	Monitor faculty research outcomes
13	Shodh Shuddhi	Ensure plagiarism-free work	Improve research quality and reputation
14	Academic Bank of Credits	Store and transfer credits	Facilitate credit redemption

Table 1.1: Summary of ICT Initiatives by the Ministry of Education

1.2 FOSSEE Project

The FOSSEE (Free/Libre and Open Source Software for Education) project promotes the use of FLOSS tools in academia and research. It is part of the National Mission on Education through Information and Communication Technology (NMEICT), Ministry of Education (MoE), Government of India.

1.2.1 Projects and Activities

The FOSSEE Project supports the use of various FLOSS tools to enhance education and research. Key activities include:

- **Textbook Companion**: Porting solved examples from textbooks using FLOSS.
- **Lab Migration**: Facilitating the migration of proprietary labs to FLOSS alternatives.
- **Niche Software Activities**: Specialized activities to promote niche software tools.
- **Forums**: Providing a collaborative space for users.
- **Workshops and Conferences**: Organizing events to train and inform users.

1.2.2 Fellowships

FOSSEE offers various internship and fellowship opportunities for students:

- Winter Internship
- Summer Fellowship
- Semester-Long Internship

Students from any degree and academic stage can apply for these internships. Selection is based on the completion of screening tasks involving programming, scientific computing, or data collection that benefit the FLOSS community. These tasks are designed to be completed within a week.

For more details, visit the official FOSSEE website.

Figure 1.1: FOSSEE Projects and Activities

1.3 Osdag Software

Osdag (Open steel design and graphics) is a cross-platform, free/libre and open-source software designed for the detailing and design of steel structures based on the Indian Standard IS 800:2007. It allows users to design steel connections, members, and systems through an interactive graphical user interface (GUI) and provides 3D visualizations of designed components. The software enables easy export of CAD models to drafting tools for construction/fabrication drawings, with optimized designs following industry best practices [1, 2, 3]. Built on Python and several Python-based FLOSS tools (e.g., PyQt and PythonOCC), Osdag is licensed under the GNU Lesser General Public License (LGPL) Version 3.

1.3.1 Osdag GUI

The Osdag GUI is designed to be user-friendly and interactive. It consists of

- **Input Dock**: Collects and validates user inputs.
- **Output Dock**: Displays design results after validation.
- **CAD Window**: Displays the 3D CAD model, where users can pan, zoom, and rotate the design.
- **Message Log**: Shows errors, warnings, and suggestions based on design checks.

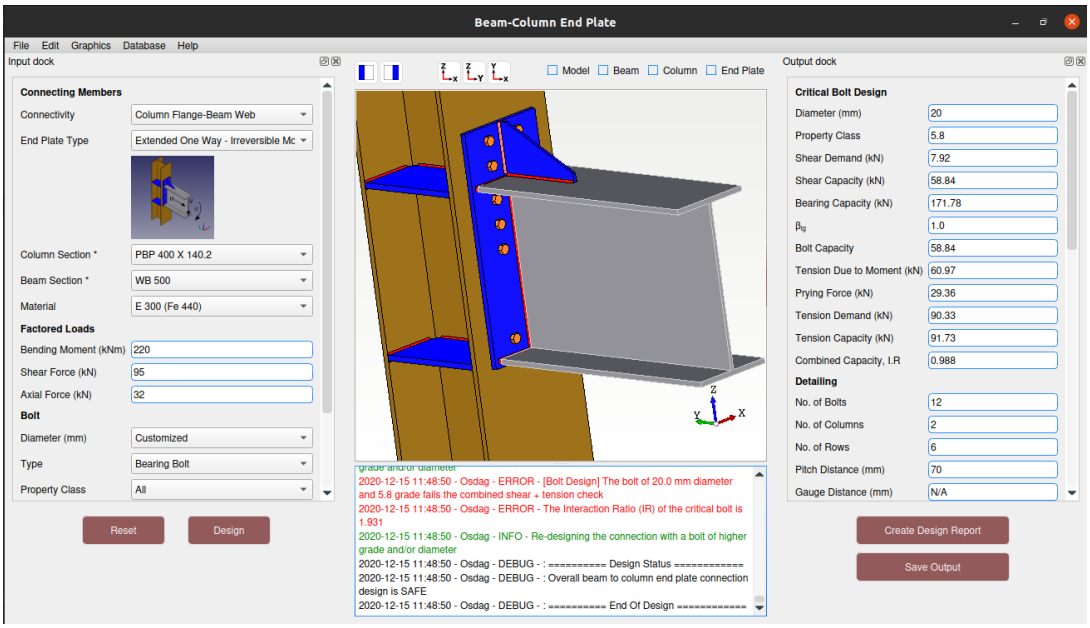

Figure 1.2: Osdag GUI

1.3.2 Features

- **CAD Model**: The 3D CAD model is color-coded and can be saved in multiple formats such as IGS, STL, and STEP.
- **Design Preferences**: Customizes the design process, with advanced users able to set preferences for bolts, welds, and detailing.
- **Design Report**: Creates a detailed report in PDF format, summarizing all checks, calculations, and design details, including any discrepancies.

For more details, visit the official Osdag website.

Chapter 2

Screening Task

2.1 Problem Statement

Develop a GUI using PyQt where different components of the Tex file are enlisted and the user can pick and select the components to print.
Project repository : **https://github.com/RoushanRaj-cmd/LatexReportGenerator**

2.2 Tasks Done

In this project, I developed a user-friendly GUI application using PyQt that allows users to load a LaTeX (.tex) file, select specific sections through an interactive checklist, preview the chosen content in real time, and generate a customized PDF report. The application automates section extraction, provides clear feedback, and streamlines the process of creating tailored technical documents, making LaTeX report generation accessible and efficient for users of all skill levels.

List of Tasks Performed

1. **Requirements Gathering and Environment Setup**
   Before starting development, I ensured all necessary software was installed. This included Python 3.6 or newer, the PyQt5 library for GUI programming, and a LaTeX distribution (like TeX Live) to handle document compilation. I also confirmed that the system had a working PDF viewer, so outputs could be checked quickly.

This setup phase ensured a smooth workflow and prevented interruptions during coding due to missing components.

2. **Designing the GUI Application**

   The application's user interface was mapped out, deciding on a clear layout that would make the user's tasks intuitive. The plan included separate panels: one for listing all the sections identified in a loaded LaTeX file, and another for real-time preview of selected content. Buttons for opening files and generating PDFs were positioned for easy access, and the window was made resizable for flexibility.

3. **Implementing File Selection and Parsing**

   File open dialog was implemented, allowing users to browse and select any '.tex' document from their computer. After loading a file, the application uses pattern matching (regular expressions) to automatically extract structured parts of the document, such as `\section{}` and `\subsection{}` commands. Each section is then dynamically shown in the GUI via checkboxes, facilitating granular selection for the final report.

4. **Building Section Selection Mechanism**

   The core feature centers around letting users pick which document parts to include. For each section listed with a checkbox, users can check or uncheck according to what they want in the final PDF. This live, modular approach gives users maximum control and encourages customization for different reporting scenarios.

5. **Content Preview Implementation**

   To help users visualize what will appear in the generated report, the application offers a live preview pane. Whenever section selections change, the preview area updates instantly to show only the currently selected content, using the original LaTeX text. This feature eliminates guesswork and ensures the desired content is included before moving to PDF creation.

6. **PDF Generation Workflow**

   After the user confirms their selection, the app constructs a new LaTeX file: it merges a template with only the chosen sections. The 'pdflatex' engine is then called in the background to generate the final PDF. The app monitors this process,

showing progress and catching any compilation errors, so users get instant feedback if something's wrong.

7. **Progress and Error Reporting**

   During PDF creation, a progress bar keeps the user informed about the operation's status. If there were issues—such as formatting problems in LaTeX, or missing dependencies—clear, actionable error messages were displayed. This feedback helped troubleshoot problems quickly and provided a smoother user experience.

8. **Custom Project Header Integration**

   To give each report a professional touch, the application automatically inserts a custom header at the top of the generated PDF. This header can include project details such as a title, author, date, or other relevant metadata, ensuring that every report looks polished and official.

9. **Polishing UI and User Experience**

   Additional refinements included making the interface easy to resize, adjusting font sizes and spacing for readability, and ensuring all controls were responsive. These adjustments made the app suitable for a wide variety of LaTeX documents and improve accessibility for all users.

10. **Documentation**

    Throughout development, all steps and features were carefully documented in the 'README.md' file. This documentation includes requirements, installation steps, troubleshooting advice, and usage instructions, enabling users (and developers) to quickly understand and make the most of the application.

2.3 Python Code

Listing 2.1: Customised Report Generator

```
1 %--------------------begin code-------------
2 import sys
3 import re
4 import os
5 import tempfile
```

```
import subprocess
from PyQt5.QtWidgets import (QApplication, QMainWindow, QWidget,
    QVBoxLayout,
                             QHBoxLayout, QLabel, QPushButton,
                                 QFileDialog,
                             QCheckBox, QScrollArea, QMessageBox,
                                 QTextEdit,
                             QSplitter, QGroupBox, QProgressBar)
from PyQt5.QtCore import Qt, QProcess

class TeXSectionSelector(QMainWindow):
    def __init__(self):
        super().__init__()
        self.initUI()
        self.tex_file = None
        self.tex_content = ""
        self.sections = []
        self.section_checkboxes = []
        self.section_content = {}
        self.preamble = ""
        self.document_env = {"begin": "", "end": ""}

    def initUI(self):
        self.setWindowTitle('TeX Section Selector')
        self.setGeometry(100, 100, 900, 600)

        main_widget = QWidget()
        self.setCentralWidget(main_widget)
        main_layout = QVBoxLayout(main_widget)

        # Top buttons
        top_layout = QHBoxLayout()
        self.open_button = QPushButton('Open .tex File')
        self.open_button.clicked.connect(self.open_tex_file)
        top_layout.addWidget(self.open_button)

        self.generate_button = QPushButton('Generate PDF')
        self.generate_button.clicked.connect(self.generate_pdf)
        self.generate_button.setEnabled(False)
```

```
42          top_layout.addWidget(self.generate_button)
43
44          main_layout.addLayout(top_layout)
45
46          # Splitter for sections and preview
47          splitter = QSplitter(Qt.Horizontal)
48
49          # Section selection area
50          section_group = QGroupBox("Sections")
51          section_layout = QVBoxLayout(section_group)
52          self.sections_area = QWidget()
53          self.sections_layout = QVBoxLayout(self.sections_area)
54          scroll = QScrollArea()
55          scroll.setWidget(self.sections_area)
56          scroll.setWidgetResizable(True)
57          section_layout.addWidget(scroll)
58
59          # Select/Deselect All buttons
60          select_buttons_layout = QHBoxLayout()
61          self.select_all_button = QPushButton('Select All')
62          self.select_all_button.clicked.connect(self.select_all_sections
                )
63          self.select_all_button.setEnabled(False)
64          select_buttons_layout.addWidget(self.select_all_button)
65
66          self.deselect_all_button = QPushButton('Deselect All')
67          self.deselect_all_button.clicked.connect(self.
                deselect_all_sections)
68          self.deselect_all_button.setEnabled(False)
69          select_buttons_layout.addWidget(self.deselect_all_button)
70
71          section_layout.addLayout(select_buttons_layout)
72
73          # Preview area
74          preview_group = QGroupBox("Preview")
75          preview_layout = QVBoxLayout(preview_group)
76          self.preview_text = QTextEdit()
77          self.preview_text.setReadOnly(True)
78          preview_layout.addWidget(self.preview_text)
```

```
79
80          # Add section and preview to splitter
81          splitter.addWidget(section_group)
82          splitter.addWidget(preview_group)
83          splitter.setSizes([300, 600])
84
85          main_layout.addWidget(splitter)
86
87          # Progress bar
88          self.progress_bar = QProgressBar()
89          self.progress_bar.setVisible(False)
90          main_layout.addWidget(self.progress_bar)
91
92          # Status bar
93          self.statusBar().showMessage('Ready')
94
95      def open_tex_file(self):
96          file_name, _ = QFileDialog.getOpenFileName(self, 'Open .tex
               file', '', 'TeX Files (*.tex)')
97          if file_name:
98              self.tex_file = file_name
99              self.statusBar().showMessage(f'Loaded: {file_name}')
100             self.parse_tex_file()
101             self.display_sections()
102             self.generate_button.setEnabled(True)
103             self.select_all_button.setEnabled(True)
104             self.deselect_all_button.setEnabled(True)
105
106     def parse_tex_file(self):
107         try:
108             with open(self.tex_file, 'r', encoding='utf-8') as file:
109                 self.tex_content = file.read()
110
111             # Extract preamble (everything before \begin{document})
112             begin_doc_match = re.search(r'\\begin{document}', self.
                   tex_content)
113             if begin_doc_match:
114                 self.preamble = self.tex_content[:begin_doc_match.start
                       ()]
```

```
                self.document_env["begin"] = begin_doc_match.group(0)
            else:
                self.preamble = ""
                self.document_env["begin"] = "\\begin{document}"

            # Extract document end
            end_doc_match = re.search(r'\\end{document}', self.
                tex_content)
            if end_doc_match:
                self.document_env["end"] = end_doc_match.group(0)
            else:
                self.document_env["end"] = "\\end{document}"

            # Extract sections
            section_pattern = r'\\(section|subsection|subsubsection)
                \{([^}]+)\}'
            section_matches = re.finditer(section_pattern, self.
                tex_content)

            self.sections = []
            prev_end = begin_doc_match.end() if begin_doc_match else 0

            for match in section_matches:
                section_type = match.group(1)
                section_title = match.group(2)
                section_start = match.start()

                # If this is the first section after \begin{document}
                if prev_end == begin_doc_match.end():
                    # Check if there's content between \begin{document}
                        and the first section
                    if section_start > prev_end:
                        preamble_content = self.tex_content[prev_end:
                            section_start]
                        if preamble_content.strip():
                            self.sections.append({
                                "type": "preamble_content",
                                "title": "Content before first section"
                                    ,
```

```
                                "start": prev_end,
                                "end": section_start
                            })

                self.sections.append({
                    "type": section_type,
                    "title": section_title,
                    "start": section_start,
                    "end": None  # Will be set later
                })
                prev_end = section_start

        # Set end positions for each section
        for i in range(len(self.sections) - 1):
            self.sections[i]["end"] = self.sections[i + 1]["start"]

        # Set the end position for the last section
        if self.sections and end_doc_match:
            self.sections[-1]["end"] = end_doc_match.start()

        # Check if there's content after the last section
        if end_doc_match and (not self.sections or self.sections
            [-1]["end"] < end_doc_match.start()):
            last_section_end = self.sections[-1]["end"] if self.
                sections else begin_doc_match.end()
            end_content = self.tex_content[last_section_end:
                end_doc_match.start()]
            if end_content.strip():
                self.sections.append({
                    "type": "end_content",
                    "title": "Content after last section",
                    "start": last_section_end,
                    "end": end_doc_match.start()
                })

        # Extract content for each section
        for section in self.sections:
            section_content = self.tex_content[section["start"]:
                section["end"]]
```

```
183                 self.section_content[section["title"]] =
                        section_content
184
185         except Exception as e:
186             QMessageBox.critical(self, "Error", f"Error parsing .tex
                    file: {str(e)}")
187             self.sections = []
188             self.section_content = {}
189
190     def display_sections(self):
191         # Clear existing checkboxes
192         for i in reversed(range(self.sections_layout.count())):
193             widget = self.sections_layout.itemAt(i).widget()
194             if widget:
195                 widget.deleteLater()
196
197         self.section_checkboxes = []
198
199         # Add checkboxes for each section
200         for section in self.sections:
201             section_type = section["type"]
202             section_title = section["title"]
203
204             # Create checkbox with appropriate indentation and
                    formatting
205             checkbox = QCheckBox()
206             checkbox.setChecked(True)  # Default to selected
207
208             if section_type == "section":
209                 checkbox.setText(f"Section: {section_title}")
210             elif section_type == "subsection":
211                 checkbox.setText(f"    Subsection: {section_title}")
212             elif section_type == "subsubsection":
213                 checkbox.setText(f"        Subsubsection: {
                        section_title}")
214             elif section_type == "preamble_content":
215                 checkbox.setText(f"[Document Introduction]")
216             elif section_type == "end_content":
217                 checkbox.setText(f"[Document Conclusion]")
```

```
218
219             checkbox.section_title = section_title
220             checkbox.section_type = section_type
221             checkbox.stateChanged.connect(self.update_preview)
222
223             self.sections_layout.addWidget(checkbox)
224             self.section_checkboxes.append(checkbox)
225
226         # Add a stretch at the end
227         self.sections_layout.addStretch()
228
229         # Update preview
230         self.update_preview()
231
232     def update_preview(self):
233         if not self.section_checkboxes:
234             return
235
236         preview_text = "Selected sections:\n\n"
237         for checkbox in self.section_checkboxes:
238             if checkbox.isChecked():
239                 preview_text += f"    {checkbox.text()}\n"
240
241                 # Add a snippet of the section content
242                 content = self.section_content.get(checkbox.
                        section_title, "")
243                 snippet = content[:200] + "..." if len(content) > 200
                        else content
244                 preview_text += f"---\n{snippet}\n---\n\n"
245             else:
246                 preview_text += f"    {checkbox.text()}\n\n"
247
248         self.preview_text.setText(preview_text)
249
250     def select_all_sections(self):
251         for checkbox in self.section_checkboxes:
252             checkbox.setChecked(True)
253
254     def deselect_all_sections(self):
```

```
255         for checkbox in self.section_checkboxes:
256             checkbox.setChecked(False)
257 
258     def fix_tex_issues(self, tex_content):
259         """Fix common issues in TeX files that might cause compilation
                errors."""
260 
261         # Fix potential issues with the Area value (incorrect decimal
                point)
262         tex_content = tex_content.replace("381.0", "3.81")
263 
264         # Fix potential issues with multirow
265         tex_content = tex_content.replace("\\multirow{14}{*}", "\\
                multirow{14}{*}{Member Properties}")
266 
267         # Fix potential issues with cline
268         tex_content = re.sub(r'\\cline{2%\n-%\n5%}', r'\\cline{2-5}',
                tex_content)
269 
270         # Fix potential issues with missing braces in textcolor command
271         tex_content = re.sub(r'\\textcolor{([^}]+)}{ *\n *\\textbf
                {([^}]*)} *\n *}',
272                              r'\\textcolor{\1}{\\textbf{\2}}',
                                 tex_content)
273 
274         # Fix potential issues with textcolor empty braces
275         tex_content = re.sub(r'\\textcolor{[^}]+}{ *\n *}', r'',
                tex_content)
276 
277         # Remove or fix empty commands
278         tex_content = re.sub(r'\\textbf{}', r'', tex_content)
279 
280         return tex_content
281 
282     def generate_pdf(self):
283         if not self.tex_file:
284             QMessageBox.warning(self, "Warning", "Please open a .tex
                    file first.")
285             return
```

```
286
287         # Get selected sections
288         selected_sections = []
289         for checkbox in self.section_checkboxes:
290             if checkbox.isChecked():
291                 selected_sections.append(checkbox.section_title)
292
293         if not selected_sections:
294             QMessageBox.warning(self, "Warning", "No sections selected.
                ")
295             return
296
297         # Create new .tex content with only selected sections
298         new_content = self.preamble + "\n" + self.document_env["begin"]
                + "\n"
299
300         for section in self.sections:
301             if section["title"] in selected_sections:
302                 new_content += self.section_content[section["title"]]
303
304         new_content += "\n" + self.document_env["end"]
305
306         # Fix common issues in the TeX content
307         new_content = self.fix_tex_issues(new_content)
308
309         # Create temporary directory for output
310         self.progress_bar.setVisible(True)
311         self.progress_bar.setValue(10)
312
313         try:
314             temp_dir = tempfile.mkdtemp()
315
316             # Create temporary .tex file
317             temp_tex_path = os.path.join(temp_dir, "selected_sections.
                tex")
318             with open(temp_tex_path, "w", encoding="utf-8") as
                temp_tex_file:
319                 temp_tex_file.write(new_content)
320
```

```
321             self.progress_bar.setValue(30)
322             self.statusBar().showMessage('Compiling PDF...')
323 
324             # Show detailed log in preview
325             self.preview_text.append("\n\nCompiling PDF...\n")
326 
327             # Run pdflatex with more verbose output
328             process = QProcess()
329             process.setWorkingDirectory(temp_dir)
330 
331             # Connect to read output
332             process.readyReadStandardOutput.connect(
333                 lambda: self.preview_text.append(str(process.
                        readAllStandardOutput().data(), 'utf-8'))
334             )
335             process.readyReadStandardError.connect(
336                 lambda: self.preview_text.append(str(process.
                        readAllStandardError().data(), 'utf-8'))
337             )
338 
339             # Run pdflatex multiple times to resolve references
340             for i in range(2):
341                 process.start("pdflatex", ["-interaction=nonstopmode",
                        temp_tex_path])
342                 process.waitForFinished(60000)  # 60 seconds timeout
343                 self.progress_bar.setValue(50 + i*20)
344 
345             # Get output file path
346             pdf_path = os.path.join(temp_dir, "selected_sections.pdf")
347 
348             if os.path.exists(pdf_path):
349                 self.progress_bar.setValue(90)
350 
351                 # Save dialog
352                 save_path, _ = QFileDialog.getSaveFileName(self, 'Save
                        PDF', '', 'PDF Files (*.pdf)')
353                 if save_path:
354                     # Copy the PDF to the selected location
355                     with open(pdf_path, "rb") as src_file:
```

```
356                         with open(save_path, "wb") as dst_file:
357                             dst_file.write(src_file.read())
358
359                     self.progress_bar.setValue(100)
360                     self.statusBar().showMessage(f'PDF saved to {
                            save_path}')
361                     QMessageBox.information(self, "Success", f"PDF
                            generated successfully and saved to {save_path}"
                            )
362                 else:
363                     self.statusBar().showMessage('PDF generation
                            cancelled')
364
365             else:
366                 error_log = os.path.join(temp_dir, "selected_sections.
                        log")
367                 error_text = "PDF compilation failed."
368
369                 if os.path.exists(error_log):
370                     with open(error_log, 'r', encoding='utf-8', errors=
                            'ignore') as log_file:
371                         error_text += "\n\nLog file:\n" + log_file.read
                                ()
372
373                 QMessageBox.critical(self, "Error", error_text)
374                 self.statusBar().showMessage('PDF compilation failed')
375
376         except Exception as e:
377             QMessageBox.critical(self, "Error", f"Error generating PDF:
                    {str(e)}")
378             self.statusBar().showMessage('PDF generation failed')
379
380         finally:
381             self.progress_bar.setVisible(False)
382
383             # Clean up temp directory if needed
384             try:
385                 if os.path.exists(temp_dir):
386                     # Keep the directory for debugging
```

```
                    self.statusBar().showMessage(f'Temporary files at:
                        {temp_dir}')
            except:
                pass

def main():
    app = QApplication(sys.argv)
    ex = TeXSectionSelector()
    ex.show()
    sys.exit(app.exec_())

if __name__ == '__main__':
    main()
%-------------------- end code ---------------
```

Task Summary Table

Task #	Task Name	Description
1	Environment Setup	Install tools, libraries, and LaTeX distribution.
2	GUI Design	Layout planning, split panels, action buttons.
3	File Handling and Parsing	Open .tex, extract sections/components.
4	Section Selection Implementation	List and enable toggling of document sections.
5	Live Content Preview	Display real-time preview of selections.
6	PDF Generation	Create and compile custom .tex to PDF.
7	Progress/Error Handling	Track progress, report errors.
8	Custom Header Integration	Insert project details/header automatically.
9	UI/UX Improvements	Enhance usability and overall UX.
10	Documentation	README updates, instructions.

Chapter 3

Reimagining Osdag Interface

3.1 Task 1: Problem Statement

The original Osdag interface, while functional, presented opportunities for improvement in terms of user experience, visual appeal, and workflow efficiency. Key areas identified for enhancement included:

- **Usability**: Streamlining input processes, improving navigation, and providing clearer feedback to users.
- **Visual Design**: Modernizing the aesthetic, improving readability, and establishing a consistent brand identity.
- **Cognitive Load**: Reducing information overload and presenting data progressively.

3.2 Task 1: Tasks Done

The redesign efforts focused on transforming the Osdag interface into a more intuitive, efficient, and visually appealing application. The key tasks undertaken include:

- **Interface Modernization**: Implemented a "Modern and Clean Interface" with a simplified visual hierarchy, moving away from a potentially cluttered or dated look.
- **Dashboard Integration**: Introduced a "Dashboard Approach" to the main screen, providing quick access to recent designs and status updates, thereby improving user

orientation and efficiency.

- **Enhanced Navigation**: Replaced the previous module selection with an improved system that emphasizes "Better Readability" and "Visual Separation," making it easier for users to navigate between different functionalities like "Connection," "Tension Member," "Flexural Member," etc.

- **Streamlined Input Forms**: Redesigned input forms to be "Streamlined," with "Better organized form elements with clear labeling," reducing user effort and potential errors.

- **Visual Branding**: Adopted a "Green Color Identity" to maintain brand consistency and made a "Choice of Darker Green" to enhance "Better Readability," "Visual Separation," "Reduced eye strain," and overall "Professional Appeal," while also giving "Logo Emphasis.

3.3 Thinking philosophy

The redesign was guided by a core philosophy centered on user needs and efficient design principles:

- **User-Centered Design**: The primary focus was on the "needs of structural engineers and steel designers," ensuring the interface truly serves its target audience.

- **Simplicity First**: Aimed to "reduce visual complexity while maintaining all functionality," preventing information overload and promoting ease of use.

- **Consistency is Key**: Emphasized "standardized UI patterns throughout the application," creating a predictable and intuitive user experience.

- **Progressive Disclosure**: Adopted a strategy to "show information only when needed to reduce cognitive load," preventing users from being overwhelmed by too much information at once.

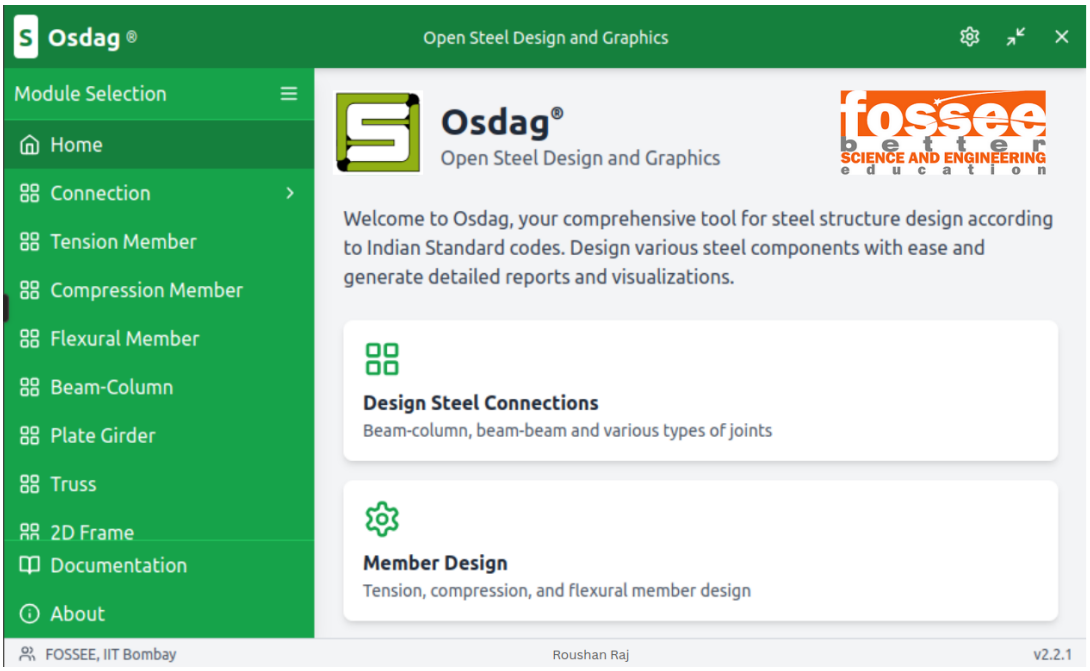

Figure 3.1: Redesigned Osdag GUI

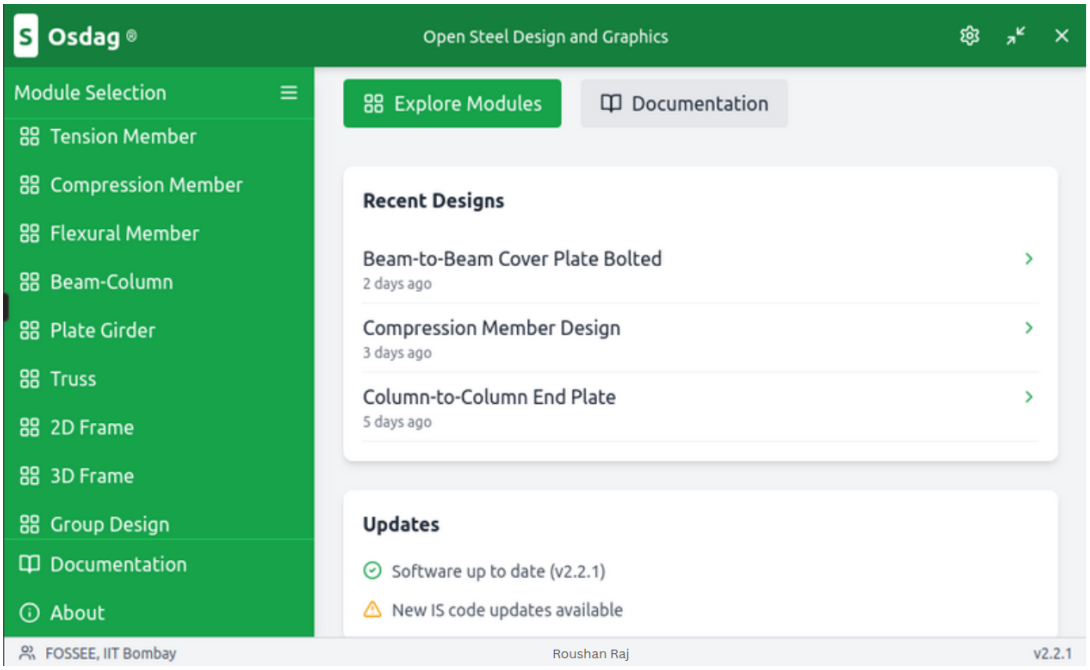

Figure 3.2: Redesigned Osdag GUI

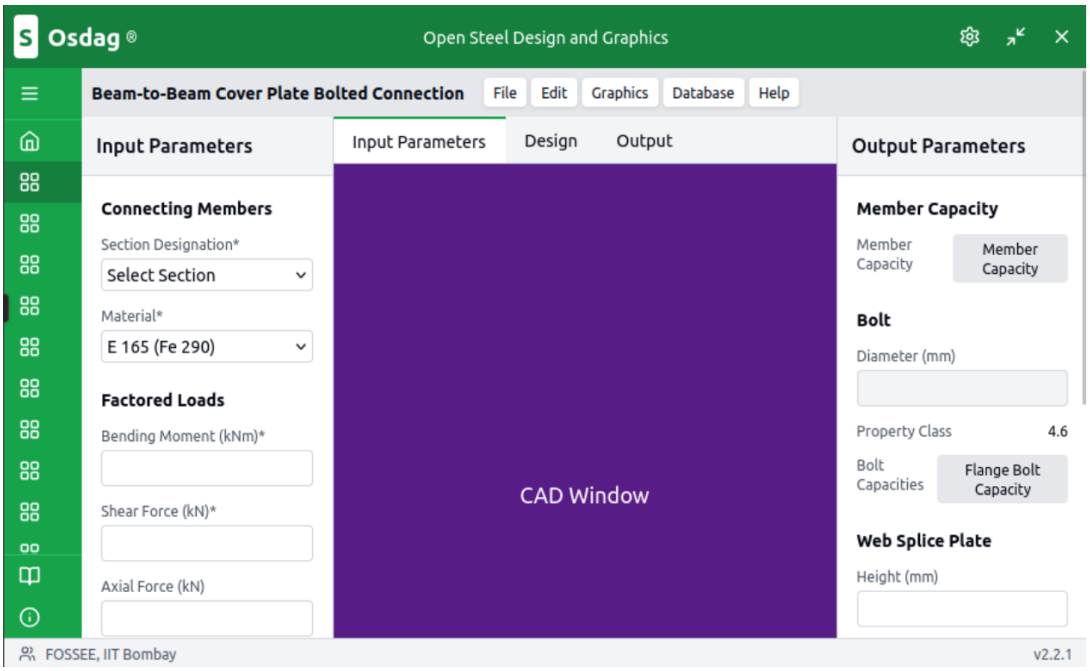

Figure 3.3: Redesigned Osdag GUI

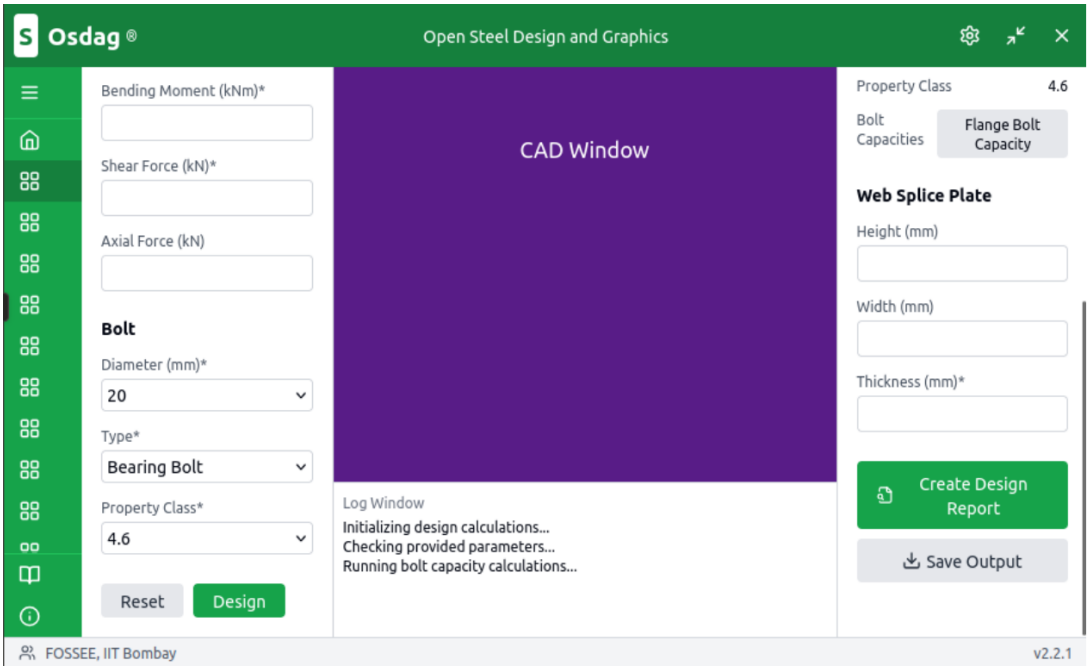

Figure 3.4: Redesigned Osdag GUI

Chapter 4

Report Generation of Welded Butt Joint

4.1 Task 1: Problem Statement

The primary objective of this task was to generate a comprehensive design report for Welded 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 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.

4.2 Task 1: Tasks Done

4.2.1 Core Save Design Function Development

- **Implemented comprehensive `save_design()` function in `butt_joint_welded.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, performs calculations, and triggers the LaTeX report compi-

lation. It includes comprehensive error handling with try-catch blocks to manage exceptions during file operations, data processing, and PDF generation.

- **Created fail-safe mechanisms with default values for missing parameters to prevent report generation failures**

  Implemented a safety net system where if any design parameter is missing or invalid, the system automatically substitutes reasonable default values rather than crashing. This involved creating a mapping of all possible design parameters with their corresponding fallback values, ensuring the report generation process is robust and user-friendly.

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

  Successfully interfaced the new welded butt joint module with Osdag's established LaTeX report generation framework. This required understanding the existing `CreateLatex` class structure, its methods, and how to properly instantiate and configure it for the specific module's needs, maintaining 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` and `pathlib` modules, ensuring the report generation works seamlessly on Windows, Linux, and macOS systems. This included handling directory separators, creating nested folder structures, and managing temporary file locations.

Report Input Dictionary Structure

- **Designed comprehensive report input dictionary containing all necessary design parameters**

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

  - Module identification and metadata

- Material properties (f_y, f_u, partial safety factors)
- Geometric parameters (plate thicknesses, widths, weld sizes)
- Design preferences (weld type, fabrication method)
- Load information (tensile force)

4.2.2 Detailed Design Verification and Check Implementation

Cover Plate Design Checks

[leftmargin=*]**Implemented cover plate thickness calculations with proper equations** Coded the mathematical logic for determining adequate cover plate thickness based on IS 800:2007 requirements:

$$\text{Single cover: } t_{cp} \geq \frac{5}{8} \times t_{min} \quad \text{(Equation 3.1)} \tag{4.1}$$

$$\text{Double cover: } t_{cp} \geq \frac{9}{16} \times t_{min} \quad \text{(Equation 3.2)} \tag{4.2}$$

Created validation checks with Pass/Fail status indicators Built a comprehensive checking system that evaluates each design criterion against IS 800:2007 requirements and provides clear Pass/Fail indicators.

Weld Design Verification System

- **Comprehensive weld size validation**

  Implemented the complete weld sizing logic from IS 800:2007, including:

  - Minimum weld size checks as per IS 800:2007 Table 21
  - Maximum weld size limits: $s_{max} = t_{min} - 1.5$

- **Weld strength calculations with proper LaTeX formatting**

  Developed weld strength analysis including:

$$\text{Design strength: } f_w = \frac{f_u}{\sqrt{3} \times \gamma_{mw}} \tag{4.3}$$

$$\text{Effective length: } L_{eff} \geq 4s \tag{4.4}$$

- **Created effective length verification ($L_{eff} \geq 4s$)**

  Coded the geometric verification system that ensures weld lengths meet minimum effectiveness requirements, accounting for end returns and practical welding limitations.

Base Metal Strength Analysis

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

  Developed parallel calculation paths for both failure modes:

$$\text{Yielding strength: } T_{dy} = \frac{A_g \times f_y}{\gamma_{m0}} \tag{4.5}$$

$$\text{Rupture strength: } T_{du} = \frac{0.9 \times A_n \times f_u}{\gamma_{m1}} \tag{4.6}$$

- **Created capacity verification with mathematical equations**

  Built the load vs. capacity comparison system with proper mathematical documentation and partial safety factor implementation.

4.2.3 Advanced LaTeX Mathematical Formatting

Mathematical Equation Integration

[leftmargin=*]**Used NoEscape LaTeX formatting for complex mathematical expressions** Mastered the PyLaTeX library's NoEscape functionality to render complex mathematical equations without Python string escaping issues, ensuring proper mathematical notation in generated reports. **Implemented proper subscript/superscript notation** Coded comprehensive mathematical notation systems using LaTeX formatting for variables like γ_{m0}, f_u, t_{cp}, ensuring professional academic standards. **Created aligned equation blocks with consistent formatting** Developed template systems for presenting multiple related equations in professionally aligned blocks with consistent spacing and mathematical flow.

Professional Report Presentation

- **Structured report in logical sections**

Designed overall report architecture with clear sectional divisions:

  - Cover Plate Design
  - Weld Design
  - Base Metal Strength Check
  - Detailing Requirements
  - Design Summary

- **Implemented color-coded Pass/Fail indicators**

  Developed visual feedback system using LaTeX color commands for immediate design status indication.

4.2.4 Design Calculations Integration

Utilization Ratio Calculations

- **Implemented comprehensive utilization tracking across all design components**

  Created monitoring system that tracks stress ratios for all critical design elements (base metal, welds, cover plates), providing insight into governing components and available capacity.

- **Created maximum utilization determination from multiple failure modes**

  Developed algorithms that evaluate all possible failure modes and identify the governing condition with highest utilization.

Detailing Requirements Implementation

- **Added construction detailing checks**

  Implemented practical construction requirements:

  - Minimum end clearance (25 mm)
  - Return weld requirements: $\max(2s, 10)$
  - Proper weld termination details

- **Implemented geometric constraint validation**

  Coded comprehensive geometric checking that verifies dimensional relationships are feasible and meet construction requirements.

4.2.5 UI and Integration Tasks

User Interface Integration

- **Modified UI template systems for report generation compatibility**

  Adapted existing user interface templates to support specific input requirements of welded butt joint report generation.

- **Updated summary popup interfaces for report parameter collection**

  Enhanced design summary popup windows to capture and display all relevant information needed for report generation.

4.3 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 func- tionality.

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 Butt Joint save_design() function

```
1 %--------------------begin code-------------
2 def save_design(self, popup_summary):
```

```
        """Save design details with proper LaTeX formatting for
            subscripts and symbols"""
        import os as os_module
        import math
        from pylatex.utils import NoEscape

        try:
            if not self.design_status:
                print("ERROR: Cannot generate report - design is not
                    complete or failed")
                return False

            def safe_get(attr, default=0):
                value = getattr(self, attr, default)
                return default if value is None else value

            self.report_input = {
                KEY_MODULE: getattr(self, 'module', 'Butt Joint Welded
                    Connection'),
                KEY_DISP_MATERIAL: safe_get('main_material', 'E 250 (Fe
                     410 W)A'),
                KEY_DISP_TENSILE_FORCE: round(safe_get('tensile_force',
                     0)/1000, 2),
                KEY_DISP_PLATE1_THICKNESS: safe_get('plate1.thickness
                    [0]', 8.0),
                KEY_DISP_PLATE2_THICKNESS: safe_get('plate2.thickness
                    [0]', 8.0),
                KEY_DISP_PLATE_WIDTH: safe_get('plates_width', 20.0),
                KEY_DISP_COVER_PLT: safe_get('cover_plate', 'Single-
                    Cover'),
                KEY_DISP_WELD_SIZE: safe_get('weld_size', 6.0),
                KEY_DISP_DP_WELD_TYPE: safe_get('weld_type', 'Shop weld
                    '),
                KEY_DISP_DP_WELD_FAB: safe_get('weld_type', 'Shop Weld'
                    ),
                KEY_DISP_DP_WELD_MATERIAL_G_O_REPORT: safe_get('fu',
                    410.0),
                KEY_DISP_ULTIMATE_STRENGTH_REPORT: safe_get('fu', 410),
                KEY_DISP_YIELD_STRENGTH_REPORT: safe_get('fy', 250),
```

```
                KEY_DISP_GAMMA_M0: safe_get('gamma_m0', 1.1),
                KEY_DISP_GAMMA_M1: safe_get('gamma_m1', 1.25),
                KEY_DISP_GAMMA_MW: safe_get('gamma_mw', 1.25)
            }

            self.report_check = []

            if self.design_status:
                plate1_thk = safe_get('plate1.thickness[0]', 8.0)
                plate2_thk = safe_get('plate2.thickness[0]', 8.0)
                t_min = min(plate1_thk, plate2_thk)
                s_min = safe_get('s_min', 3)
                s_max = safe_get('s_max', 6.5)
                weld_size = safe_get('weld_size', 6)
                fu = safe_get('fu', 410.0)
                fy = safe_get('fy', 250)
                gamma_mw = safe_get('gamma_mw', 1.25)
                gamma_m0 = safe_get('gamma_m0', 1.1)
                gamma_m1 = safe_get('gamma_m1', 1.25)
                tensile_force = safe_get('tensile_force', 10000)
                plates_width = safe_get('plates_width', 20.0)
                cover_plate = safe_get('cover_plate', 'Single-Cover')

                f_w = fu / (math.sqrt(3) * gamma_mw)
                N_f = 2 if "double" in cover_plate.lower() else 1
                te = 0.707 * weld_size

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

                if "double" in cover_plate.lower():
                    tcp_req = (9.0/16.0) * t_min
                    tcp_formula = NoEscape(f"\\small t$_{{cp}}$ $\\geq$
                        9/16 $\\times$ t$_{{min}}$ $\\geq$ 9/16 $\\
                       times$ {t_min} $\\geq$ {round(tcp_req, 2)}~mm")
                else:
                    tcp_req = (5.0/8.0) * t_min
```

```
                    tcp_formula = NoEscape(f"\\small t$_{{cp}}$ $\\geq$
                        5/8 $\\times$ t$_{{min}}$ $\\geq$ 5/8 $\\times$
                        {t_min} $\\geq$ {round(tcp_req, 2)}~mm")

                tcp_provided = safe_get('
                    calculated_cover_plate_thickness', 8.0)
                t1 = ('Cover Plate Thickness', tcp_formula, f'{
                    tcp_provided} mm', 'Pass' if tcp_provided >= tcp_req
                     else 'Fail')
                self.report_check.append(t1)

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

                t1 = ('Minimum Weld Size',
                    NoEscape(f'\\small t$_{{w,min}}$ based on thinner
                        part = max({plate1_thk}, {plate2_thk})~s$_{{min
                        }}$ based on thicker part = {s_min}~[Ref.~IS
                        ~800:2007, Table~21, Cl.10.5.2.3]'),
                    f'{weld_size} mm',
                    'Pass' if weld_size >= s_min else 'Fail')
                self.report_check.append(t1)

                t1 = ('Maximum Weld Size',
                    NoEscape(f'\\small Thickness of thinner part = min
                        ({plate1_thk}, {plate2_thk}) = {t_min}~s$_{{max
                        }}$ = {s_max}~[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(t1)

                t1 = ('Weld Design Strength',
                    NoEscape(f'\\small f$_{{w}}$ = f$_{{u}}$/$\\sqrt
                        {{3}}$ $\\times$ $\\gamma$$_{{mw}}$ = {fu}/$\\
                        sqrt{{3}}$ $\\times$ {gamma_mw} = {round(f_w, 2)
                        }~N/mm$^2$'),
                    f'{round(f_w, 2)} N/mm  ',
```

```
92                      'Pass')
93                  self.report_check.append(t1)
94
95                  L_req = tensile_force / (N_f * te * f_w) if (N_f * te *
                        f_w) > 0 else 0
96
97                  t1 = ('Required Weld Length',
98                      NoEscape(f'\\small L$_{{req}}$ = P/(N$_{{f}}$ $\\
                            times$ t$_{{e}}$ $\\times$ f$_{{w}}$) = {round(
                            tensile_force, 1)}/({N_f} $\\times$ {round(te,
                            2)} $\\times$ {round(f_w, 2)}) = {round(L_req,
                            1)}~mm'),
99                      f'{plates_width} mm',
100                     'Pass' if plates_width >= L_req else 'Fail')
101                 self.report_check.append(t1)
102
103                 L_eff_min = 4 * weld_size
104                 L_eff_provided = plates_width - (2 * weld_size)
105
106                 t1 = ('Effective Weld Length',
107                     NoEscape(f'\\small L$_{{eff}}$ $\\geq$ 4s = 4 $\\
                            times$ {weld_size} = {L_eff_min}~mm~[DDCL~3.5]')
                            ,
108                     f'{round(L_eff_provided, 0)} mm',
109                     'Pass' if L_eff_provided >= L_eff_min else 'Fail')
110                 self.report_check.append(t1)
111
112                 weld_strength = N_f * te * L_eff_provided * f_w
113
114                 t1 = ('Weld Strength Verification',
115                     NoEscape(f'\\small P$_{{weld}}$ = N$_{{f}}$ $\\
                            times$ t$_{{e}}$ $\\times$ L$_{{eff}}$ $\\times$
                             f$_{{w}}$ = {N_f} $\\times$ {round(te, 2)} $\\
                            times$ {round(L_eff_provided, 1)} $\\times$ {
                            round(f_w, 2)} = {round(weld_strength, 0)}~N'),
116                     f'{round(weld_strength/1000, 2)} kN',
117                     'Pass' if weld_strength >= tensile_force else 'Fail
                            ')
118                 self.report_check.append(t1)
```

```
119
120                 # Base Metal Strength Check Section
121                 t1 = ('SubSection', 'Base Metal Strength Check', '|p{4
                        cm}|p{6cm}|p{4cm}|p{2cm}|')
122                 self.report_check.append(t1)
123
124                 A_g = t_min * plates_width
125                 A_n = A_g
126                 T_dy = A_g * fy / gamma_m0
127
128                 t1 = ('Yielding Strength',
129                     NoEscape(f'\\small As per Cl.~6.2, IS~800:2007 T$_
                            {{dy}}$ = A$_{{g}}$ $\\times$ f$_{{y}}$/$\\
                            gamma$$_{{m0}}$ = {A_g} $\\times$ {fy}/{gamma_m0
                            } = {round(T_dy/1000, 2)}~kN'),
130                     f'{round(T_dy/1000, 2)} kN',
131                     'Pass')
132                 self.report_check.append(t1)
133
134                 T_du = 0.9 * A_n * fu / gamma_m1
135
136                 t1 = ('Rupture Strength',
137                     NoEscape(f'\\small As per Cl.~6.3, IS~800:2007 T$_
                            {{du}}$ = 0.9 $\\times$ A$_{{n}}$ $\\times$ f$_
                            {{u}}$/$\\gamma$$_{{m1}}$ = 0.9 $\\times$ {A_n}
                            $\\times$ {fu}/{gamma_m1} = {round(T_du/1000, 2)
                            }~kN'),
138                     f'{round(T_du/1000, 2)} kN',
139                     'Pass')
140                 self.report_check.append(t1)
141
142                 T_db = min(T_dy, T_du)
143
144                 t1 = ('Base Metal Capacity',
145                     NoEscape(f'\\small T$_{{db}}$ = min(T$_{{dy}}$, T$_
                            {{du}}$) = min({round(T_dy/1000, 2)}, {round(
                            T_du/1000, 2)}) = {round(T_db/1000, 2)}~kN~[Ref
                            :~IS~800:2007, Cl.6.2, 6.3]'),
146                     f'{round(T_db/1000, 2)} kN',
```

```
                    'Pass')
                self.report_check.append(t1)

                # Detailing Requirements Section
                t1 = ('SubSection', 'Detailing Requirements', '|p{4cm}|
                    p{6cm}|p{4cm}|p{2cm}|')
                self.report_check.append(t1)

                t1 = ('Minimum End Clearance',
                    NoEscape('\\small End clearance $\\geq$ 25~mm~[DDCL
                        ~3.8]'),
                    '25 mm provided',
                    'Pass')
                self.report_check.append(t1)

                return_weld_min = max(2 * weld_size, 10)
                t1 = ('Return Weld Length',
                    NoEscape(f'\\small Return weld $\\geq$ max(2s, 10)
                        = max({2*weld_size}, 10) = {return_weld_min}~mm
                        ~[IS~800:2007, Cl.~10.5.10.2]'),
                    f'{return_weld_min} mm provided',
                    'Pass')
                self.report_check.append(t1)

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

                overall_capacity = min(weld_strength, T_db)
                t1 = ('Overall Capacity',
                    NoEscape(f'\\small min(Weld Capacity, Base Metal
                        Capacity) = min({round(weld_strength/1000, 2)},
                        {round(T_db/1000, 2)}) = {round(overall_capacity
                        /1000, 2)}~kN'),
                    f'{round(overall_capacity/1000, 2)} kN',
                    'Pass')
                self.report_check.append(t1)

```

```
178                 utilization_ratio = tensile_force / overall_capacity if
                        overall_capacity > 0 else 0
179                 t1 = ('Utilization Ratio',
180                     NoEscape(f'\\small UR = Applied Force / Design
                            Capacity $\\leq$ 1.0 UR = P$_{{N}}$/min(C$_{{w}}
                            $, T$_{{db}}$) = {round(tensile_force, 1)}/min({
                            round(weld_strength, 0)}, {round(T_db, 0)}) = {
                            round(utilization_ratio, 3)}'),
181                     f'{round(utilization_ratio, 3)}',
182                     'Pass' if utilization_ratio <= 1.0 else 'Fail')
183                 self.report_check.append(t1)
184
185                 overall_pass = (tensile_force <= overall_capacity and
186                                 utilization_ratio <= 1.0 and
187                                 weld_size >= s_min and
188                                 weld_size <= s_max and
189                                 L_eff_provided >= L_eff_min)
190
191                 t1 = ('Overall Design Status',
192                     'All design checks must pass',
193                     'Connection is SAFE for applied loads' if
                            overall_pass else 'Connection FAILS',
194                     'Pass' if overall_pass else 'Fail')
195                 self.report_check.append(t1)
196
197             else:
198                 t1 = ('SubSection', 'Design Status', '|p{4cm}|p{6cm}|p
                        {4cm}|p{2cm}|')
199                 self.report_check.append(t1)
200                 t1 = ('Design Status', 'Design calculation failed or
                        not performed', 'Design Fails', 'Fail')
201                 self.report_check.append(t1)
202
203             required_fields = {
204                 'ProjectTitle': 'Welded Butt Joint Design Report',
205                 'Subtitle': 'Structural Steel Connection Design',
206                 'JobNumber': 'JOB-001',
207                 'Client': 'Client Name',
208                 'AdditionalComments': 'Design completed successfully.
```

```
                    All design checks conform to IS 800:2007 provisions.
                     Connection is safe for the applied loads.',
209                 'ProfileSummary': {
210                     'CompanyName': 'Engineering Consultant',
211                     'CompanyLogo': '',
212                     'Group/TeamName': 'Structural Design Team',
213                     'Designer': 'Design Engineer'
214                 },
215                 'does_design_exist': self.design_status,
216                 'logger_messages': 'INFO: Design completed successfully
                    '
217             }
218
219             for field, default_value in required_fields.items():
220                 if field not in popup_summary or not popup_summary[
                        field]:
221                     popup_summary[field] = default_value
222
223             if 'ProfileSummary' not in popup_summary:
224                 popup_summary['ProfileSummary'] = {}
225             for key, value in required_fields['ProfileSummary'].items()
                    :
226                 if key not in popup_summary['ProfileSummary']:
227                     popup_summary['ProfileSummary'][key] = value
228
229             fname_no_ext = popup_summary.get('filename', '
                    welded_butt_joint_comprehensive_report')
230             fname_no_ext = os_module.path.basename(fname_no_ext)
231             folder = popup_summary.get('folder', './reports')
232
233             os_module.makedirs(folder, exist_ok=True)
234
235             try:
236                 from ...design_report.reportGenerator_latex import
                        CreateLatex
237                 latex = CreateLatex()
238
239                 Disp_2d_image = []
240                 Disp_3D_image_path = ""
```

```
241
242                 result = latex.save_latex(
243                     self.report_input,
244                     self.report_check,
245                     popup_summary,
246                     fname_no_ext,
247                     folder,
248                     Disp_2d_image,
249                     Disp_3D_image_path,
250                     getattr(self, 'module', 'ButtJointWelded')
251                 )
252
253                 pdf_file_path = os_module.path.join(folder, f"{
                        fname_no_ext}.pdf")
254                 if os_module.path.exists(pdf_file_path):
255                     file_size = os_module.path.getsize(pdf_file_path)
256                     print(f"SUCCESS: Report generated: {pdf_file_path}
                            ({file_size} bytes)")
257                     return True
258                 else:
259                     print("ERROR: PDF not found")
260                     return False
261
262             except Exception as e:
263                 print(f"ERROR: {e}")
264                 import traceback
265                 traceback.print_exc()
266                 return False
267
268         except Exception as e:
269             print(f"CRITICAL ERROR: {e}")
270             return False
271
272         """Save design details for report generation"""
273
274         # Report input dictionary
275         self.report_input = {
276             KEY_MODULE: self.module,
277             KEY_MAIN_MODULE: self.mainmodule,
```

```
278
279             # Connection details
280             KEY_DISP_AXIAL: round(self.tensile_force/1000, 2),  #
                    Convert N to kN
281
282             # Connecting Members
283             "Connecting Members": "TITLE",
284             KEY_DISP_PLATETHK: str([int(d) for d in [self.plate1.
                    thickness[0], self.plate2.thickness[0]]]),
285             KEY_DISP_MATERIAL: self.main_material,
286             KEY_DISP_ULTIMATE_STRENGTH_REPORT: self.plate1.fu,
287             KEY_DISP_YIELD_STRENGTH_REPORT: self.plate1.fy,
288             KEY_DISP_PLATE_WIDTH: self.plates_width,
289
290             # Weld Details
291             "Weld Details - Input and Design Preference": "TITLE",
292             KEY_DISP_DP_WELD_TYPE: self.weld_type,
293             KEY_DISP_DP_WELD_FAB: self.weld.fabrication,
294             KEY_DISP_DP_WELD_MATERIAL_G_O_REPORT: self.weld.fu,
295             KEY_DISP_WELD_SIZE: self.weld_size,
296
297             # Safety Factors
298             "Safety Factors": "TITLE",
299             KEY_DISP_GAMMA_M0: self.gamma_m0,
300             KEY_DISP_GAMMA_M1: self.gamma_m1,
301             KEY_DISP_GAMMA_MW: self.gamma_mw
302         }
303
304         self.report_check = []
305
306         # Selected Member Data
307         t1 = ('Selected', 'Selected Member Data', '|p{5cm}|p{2cm}|p{2cm
                }|p{2cm}|p{4cm}|')
308         self.report_check.append(t1)
309
310         if self.design_status:
311             # Member Check
312             t1 = ('SubSection', 'Member Check', '|p{2.5cm}|p{4.5cm}|p
                    {7.5cm}|p{1cm}|')
```

```
313             self.report_check.append(t1)
314
315             t1 = (KEY_DISP_TENSION_YIELDCAPACITY, '',
316                   cl_6_2_tension_yield_capacity_member(l=None, t=None,
                          f_y=self.plate1.fy, gamma=self.gamma_m0,
317                                                       T_dg=round(self.
                                                          T_db/1000, 2),
                                                          area=self.A_g),
                                                           '')
318             self.report_check.append(t1)
319
320             # Weld Design
321             t1 = ('SubSection', 'Weld Design', '|p{3cm}|p{6.5cm}|p{5cm
                    }|p{1cm}|')
322             self.report_check.append(t1)
323
324             t1 = (DISP_MIN_WELD_SIZE,
325                   cl_10_5_2_3_min_fillet_weld_size_required(self.
                          weld_connecting_plates, self.weld.min_weld, self.
                          weld.red),
326                   display_prov(self.weld_size, "s"),
327                   get_pass_fail(self.weld.min_weld, self.weld_size,
                          relation="leq"))
328             self.report_check.append(t1)
329
330             t1 = (DISP_MAX_WELD_SIZE,
331                   cl_10_5_3_1_max_weld_size(self.weld_connecting_plates
                          , self.weld_size_max),
332                   display_prov(self.weld_size, "s"),
333                   get_pass_fail(self.weld_size, self.weld_size_max,
                          relation="leq"))
334             self.report_check.append(t1)
335
336             t1 = (DISP_THROAT,
337                   cl_10_5_3_1_throat_thickness_req(),
338                   cl_10_5_3_1_throat_thickness_weld(self.weld_size,
                          self.Kt),
339                   get_pass_fail(3.0, self.weld_size, relation="leq"))
340             self.report_check.append(t1)
```

```
341
342             t1 = (DISP_EFF, "",
343                   display_prov(self.weld_length_effective, "l_w"), "")
344             self.report_check.append(t1)
345
346             t1 = (DISP_WELD_STRENGTH,
347                   weld_strength_req(V=0.0, A=self.tensile_force, M=0.0,
                          Ip_w=1.0,
348                                     y_max=0.0, x_max=0.0, l_w=self.
                                        weld_length_effective,
349                                     R_w=self.weld.stress),
350                   cl_10_5_7_1_1_weld_strength(weld_conn_plates_fu=[self
                        .fu], gamma_mw=self.gamma_mw,
351                                               t_t=round(self.weld.throat,
                                                    2),
352                                               f_w=round(self.weld.
                                                   strength, 2)),
353                   get_pass_fail(self.weld.stress, self.weld.strength,
                        relation="leq"))
354             self.report_check.append(t1)
355
356             # Long joint check if applicable
357             if hasattr(self, 'beta_L'):
358                 t1 = (KEY_OUT_LONG_JOINT_WELD, long_joint_welded_req(),
359                       cl_10_5_7_3_weld_strength_post_long_joint(h=self.
                            plates_width,
360                                                                 l=self.
                                                                     weld_length_provide
                                                                     ,
361                                                                 t_t=self.
                                                                     weld.
                                                                     throat
                                                                     ,
362                                                                 ws=self.
                                                                     weld.
                                                                     strength
                                                                     ,
363                                                                 wsr=self.
                                                                     weld.
```

```
                                                                    strength_red
                                                                    ), "")
364                 self.report_check.append(t1)
365
366                 t1 = (KEY_OUT_DISP_RED_WELD_STRENGTH,
367                       display_prov(round(self.weld.stress, 2), "f_w"),
368                       display_prov(round(self.weld.strength_red, 2), "
                          f_wd"),
369                       get_pass_fail(self.weld.stress, self.weld.
                          strength_red, relation="leq"))
370                 self.report_check.append(t1)
371
372             # Final Checks
373             t1 = ('SubSection', 'Capacity Checks', '|p{3.5cm}|p{4.5cm}|
                    p{6cm}|p{1.5cm}|')
374             self.report_check.append(t1)
375
376             t1 = ('Base Metal Strength (kN)',
377                   display_prov(round(self.tensile_force/1000, 2), "P"),
378                   display_prov(round(self.T_db/1000, 2), "T_db"),
379                   get_pass_fail(self.tensile_force, self.T_db, relation
                      ="leq"))
380             self.report_check.append(t1)
381
382             t1 = ('Overall Utilization Ratio',
383                   required_IR_or_utilisation_ratio(IR=1),
384                   display_prov(round(self.utilization_ratio, 3), "IR"),
385                   get_pass_fail(self.utilization_ratio, 1, relation="
                      leq"))
386             self.report_check.append(t1)
387
388         else:
389             t1 = ('SubSection', 'Design Status', '|p{3.5cm}|p{4.5cm}|p
                    {6cm}|p{1.5cm}|')
390             self.report_check.append(t1)
391             t1 = ('Design Status', '', 'Design Fails', 'Fail')
392             self.report_check.append(t1)
393
394         # Images
```

```
395         Disp_2d_image = []
396         Disp_3D_image = "/ResourceFiles/images/3d.png"
397 
398         rel_path = os.path.abspath(".")
399         rel_path = rel_path.replace("\\", "/")
400 
401         fname_no_ext = popup_summary['filename']
402 
403         CreateLatex.save_latex(CreateLatex(), self.report_input, self.
                report_check, popup_summary,
404                                fname_no_ext, rel_path, Disp_2d_image,
                                   Disp_3D_image, module=self.module)
405 
406 %-------------------- end code ---------------
```

4.3.1 Explanation of the Code 1

The save_design function serves as the comprehensive report generation engine for the welded butt joint module that systematically transforms design calculations into professional PDF documentation. It begins by validating that the design was completed successfully, then safely collects all design parameters using fail-safe mechanisms to prevent crashes from missing data. The function performs a sequential series of engineering verification checks following IS 800:2007 standards - starting with cover plate thickness validation, progressing through weld size and strength calculations, base metal capacity analysis, construction detailing requirements, and culminating in an overall design adequacy assessment with utilization ratios. Each verification step is mathematically calculated, formatted with proper LaTeX notation including subscripts and engineering symbols, assigned a Pass/Fail status, and organized into a structured report format. Finally, the function compiles all this organized data through the existing Osdag LaTeX infrastructure to generate a publication-ready PDF report complete with professional formatting, code references, mathematical equations, and clear visual indicators of design compliance, while providing comprehensive error handling and file management throughout the entire process to ensure robust operation.

Listing 4.2: Welded Butt Joint ui_summary_popup.py

```
%--------------------begin code-------------
def pdf_file_message(self, filename):
        """Handle PDF generation completion and display appropriate
           message"""
        try:
            # **FIX**: Determine the correct UI object reference
            ui_ref = None
            if hasattr(self, 'ui') and self.ui:
                ui_ref = self.ui
            elif hasattr(self, 'lblmessage'):
                ui_ref = self  # self is the UI object directly
            else:
                print("Could not find UI reference")
                return

            # **FIX**: Handle encoding issues when reading log file
            logfile_path = filename + '.log'
            logs = ""

            if os.path.exists(logfile_path):
                # Try multiple encoding strategies to handle LaTeX log
                   files
                encodings_to_try = ['utf-8', 'latin-1', 'cp1252', 'iso
                   -8859-1']

                for encoding in encodings_to_try:
                    try:
                        with open(logfile_path, 'r', encoding=encoding)
                             as logfile:
                            logs = logfile.read()
                        print(f"Successfully read log file with {
                           encoding} encoding")
                        break
                    except UnicodeDecodeError:
                        continue
                    except Exception as e:
                        print(f"Error reading with {encoding}: {e}")
                        continue
                else:
```

```
35                      # If all encodings fail, try with error handling
36                      try:
37                          with open(logfile_path, 'r', encoding='utf-8',
                                errors='ignore') as logfile:
38                              logs = logfile.read()
39                          print("Read log file with UTF-8 and ignored
                                decode errors")
40                      except Exception as e:
41                          print(f"Final fallback failed: {e}")
42                          logs = "Could not read log file due to encoding
                                issues"
43              else:
44                  logs = "Log file not found"
45                  print(f"Log file not found: {logfile_path}")
46
47              # Check for LaTeX errors in the log content
48              has_latex_error = False
49              has_fatal_error = False
50
51              if logs:
52                  # Check for various types of LaTeX errors
53                  error_indicators = [
54                      "LaTeX Error:",
55                      "! LaTeX Error:",
56                      "Fatal error occurred",
57                      "Emergency stop",
58                      "! ==> Fatal error occurred",
59                      "Runaway argument",
60                      "! Undefined control sequence",
61                      "! Missing $ inserted",
62                      "! File ended while scanning"
63                  ]
64
65                  for error in error_indicators:
66                      if error in logs:
67                          has_latex_error = True
68                          if "Fatal" in error or "Emergency stop" in
                                error:
69                              has_fatal_error = True
```

```
                    break

            # Check if PDF was actually generated regardless of log errors
            pdf_file_path = filename + '.pdf'
            pdf_exists = os.path.exists(pdf_file_path)

            # Determine the appropriate message
            if pdf_exists:
                if has_fatal_error:
                    message = "<font color='orange'>PDF generated with warnings. Please check the output.</font>"
                elif has_latex_error:
                    message = "<font color='orange'>PDF generated successfully with some LaTeX warnings.</font>"
                else:
                    message = "<font color='green'>PDF generated successfully.</font>"
            else:
                # PDF not generated
                if has_latex_error or has_fatal_error:
                    message = "<font color='red'>ERROR: LaTeX file processing failed. PDF not generated.</font>"
                else:
                    message = "<font color='red'>ERROR: PDF generation failed for unknown reason.</font>"

            # **FIX**: Set message using the correct UI reference
            if hasattr(ui_ref, 'lblmessage'):
                ui_ref.lblmessage.setText(message)
            elif hasattr(ui_ref, 'label') and hasattr(ui_ref.label, 'setText'):
                ui_ref.label.setText(message)
            else:
                print(f"Message would be: {message}")

            # **FIX**: Enable/disable buttons using correct UI reference
            if pdf_exists:
```

```
                # Enable view buttons if they exist
                for btn_name in ['btn_viewPDF', 'pushButton_3', '
                    pushButton_view']:
                    if hasattr(ui_ref, btn_name):
                        getattr(ui_ref, btn_name).setEnabled(True)
            else:
                # Disable view buttons
                for btn_name in ['btn_viewPDF', 'pushButton_3', '
                    pushButton_view']:
                    if hasattr(ui_ref, btn_name):
                        getattr(ui_ref, btn_name).setEnabled(False)

            # Update progress bar if it exists
            for progress_name in ['progressBar', 'progress']:
                if hasattr(ui_ref, progress_name):
                    getattr(ui_ref, progress_name).setValue(100)
                    break

            # Enable close buttons
            for btn_name in ['btn_close', 'buttonBox']:
                if hasattr(ui_ref, btn_name):
                    getattr(ui_ref, btn_name).setEnabled(True)

            print(f"PDF exists: {pdf_exists}")
            print(f"Has LaTeX error: {has_latex_error}")
            print(f"Has fatal error: {has_fatal_error}")

        except Exception as e:
            print(f"Error in pdf_file_message: {e}")
            import traceback
            traceback.print_exc()

            # Fallback message
            pdf_file_path = filename + '.pdf'
            if os.path.exists(pdf_file_path):
                message = "<font color='green'>PDF generated
                    successfully.</font>"
            else:
                message = "<font color='red'>ERROR: Could not determine
```

```
                    PDF generation status.</font>"
137
138             # Try to set message with fallback approach
139             try:
140                 if hasattr(self, 'lblmessage'):
141                     self.lblmessage.setText(message)
142                 elif hasattr(self, 'ui') and hasattr(self.ui, '
                        lblmessage'):
143                     self.ui.lblmessage.setText(message)
144                 else:
145                     print(f"Fallback message: {message}")
146             except:
147                 print(f"Could not set UI message: {message}")
148 %-------------------- end code ---------------
```

4.3.2 Explanation of the Code 2

The pdf_file_message function serves as an intelligent PDF generation completion handler that provides comprehensive status analysis by first resolving UI object references across different dialog structures, then performing robust log file analysis using multiple encoding strategies to handle international characters, intelligently detecting various types of LaTeX errors (warnings vs. fatal errors), determining nuanced status messages with color-coded HTML formatting based on both file existence and compilation quality, dynamically updating UI elements (enabling/disabling view buttons, updating progress bars, managing close buttons) according to the generation results, and implementing comprehensive fallback mechanisms that ensure users always receive meaningful status feedback even when detailed analysis encounters errors, making it a critical component for user experience in the report generation workflow.

4.3.3 Full code integration

Here is the main ui code i.e. ui_template.py containing button to generate design report. The code of pop_up window that appears for the report is there in the ui_summary_popup.py and is integrated here with the ui_template.py file. And the main logic of report generation resides in the save_design() function which is present in src/osdag/connection/butt_joint_welded.py

.

Listing 4.3: Welded Butt Joint ui_template.py

```
%--------------------begin code-------------
        if main.design_status:
            try:
                from ..osdagMainSettings import backend_name
                off_display, _, _, _ = init_display_off_screen(
                    backend_str=backend_name())
                print('off_display', off_display)

                # Check if commLogicObj exists and is properly
                    initialized
                if hasattr(self, 'commLogicObj') and self.commLogicObj
                    is not None:
                    # Store original display settings
                    original_display = self.commLogicObj.display
                    original_component = getattr(self.commLogicObj, '
                        component', None)

                    # Set up for image generation
                    self.commLogicObj.display = off_display
                    self.commLogicObj.display_3DModel("Model", "
                        gradient_bg")

                    image_folder_path = "./ResourceFiles/images"
                    if not os.path.exists(image_folder_path):
                        os.makedirs(image_folder_path)

                    off_display.set_bg_gradient_color
                        ([255,255,255],[255,255,255])
                    off_display.ExportToImage(os.path.join(
                        image_folder_path, '3d.png'))
                    off_display.View_Front()
                    off_display.FitAll()
                    off_display.ExportToImage(os.path.join(
                        image_folder_path, 'front.png'))
                    off_display.View_Top()
                    off_display.FitAll()
```

```
29                      off_display.ExportToImage(os.path.join(
                            image_folder_path, 'top.png'))
30                      off_display.View_Right()
31                      off_display.FitAll()
32                      off_display.ExportToImage(os.path.join(
                            image_folder_path, 'side.png'))
33
34                      # Restore original display settings
35                      self.commLogicObj.display = original_display
36                      if original_component is not None:
37                          self.commLogicObj.component =
                                original_component
38
39                      print("3D images generated successfully")
40                  else:
41                      print("commLogicObj not available - skipping 3D
                            image generation")
42                      # Create default/placeholder images directory
43                      image_folder_path = "./ResourceFiles/images"
44                      if not os.path.exists(image_folder_path):
45                          os.makedirs(image_folder_path)
46
47              except Exception as e:
48                  print(f"Error generating 3D images: {str(e)}")
49                  # Ensure images directory exists even if image
                        generation fails
50                  image_folder_path = "./ResourceFiles/images"
51                  if not os.path.exists(image_folder_path):
52                      os.makedirs(image_folder_path)
53
54          # Open the summary popup dialog
55          self.new_window = QtWidgets.QDialog(self)
56          self.new_ui = Ui_Dialog1(main.design_status,loggermsg=self.
                textEdit.toPlainText())
57          self.new_ui = Ui_Dialog1(main.design_status, loggermsg=self.
                textEdit.toPlainText())
58
59      def generate_design_report(self, main, dialog_window):
60          """Generate the actual design report"""
```

```
61          try:
62              # Collect all inputs from the popup
63              popup_summary = self.getPopUpInputs()
64
65              # Add additional required fields
66              popup_summary['filename'] = popup_summary.get('filename', f
                    '{main.module_name(main).replace(" ", "_")}_report')
67              popup_summary['folder'] = popup_summary.get('folder', './
                    reports')
68              popup_summary['logger_messages'] = self.textEdit.
                    toPlainText()
69              popup_summary['does_design_exist'] = main.design_status
70
71              # **CRITICAL FIX**: Get the actual design instance, not the
                     class
72              design_instance = None
73
74              # Method 1: Try to get instance from commLogicObj if it
                    exists
75              if hasattr(self, 'commLogicObj') and self.commLogicObj is
                    not None:
76                  if hasattr(self.commLogicObj, 'design_obj') and self.
                        commLogicObj.design_obj is not None:
77                      design_instance = self.commLogicObj.design_obj
78                      print("DEBUG: Using design instance from
                            commLogicObj")
79
80              # Method 2: If no instance found, create one and set it up
                    with design inputs
81              if design_instance is None:
82                  print("DEBUG: Creating new design instance")
83                  design_instance = main()  # Create instance from class
84
85                  # **FIX**: Copy ALL attributes from the class to the
                        instance
86                  for attr_name in dir(main):
87                      if not attr_name.startswith('_') and not callable(
                            getattr(main, attr_name)):
88                          try:
```

```
                            attr_value = getattr(main, attr_name)
                            if attr_value is not None:
                                setattr(design_instance, attr_name,
                                    attr_value)
                        except:
                            pass

                # Set the design inputs on the instance if available
                if hasattr(self, 'design_inputs') and self.
                    design_inputs:
                    for key, value in self.design_inputs.items():
                        if hasattr(design_instance, key):
                            setattr(design_instance, key, value)

                # Copy design status from the class
                design_instance.design_status = getattr(main, '
                    design_status', True)

                # Copy design status from the class
                if hasattr(main, 'design_status'):
                    design_instance.design_status = main.design_status
                else:
                    design_instance.design_status = True  # Assume
                        successful if no status

                # Run the design calculation if needed
                if hasattr(design_instance, 'design_status') and not
                    design_instance.design_status:
                    print("DEBUG: Running design calculation on
                        instance")
                    try:
                        if hasattr(design_instance, 'design_connection'
                            ):
                            design_instance.design_connection()
                        elif hasattr(design_instance, 'design'):
                            design_instance.design()
                    except Exception as e:
                        print(f"DEBUG: Error running design: {e}")

```

```
121             # Now call save_design on the actual instance
122             if design_instance and hasattr(design_instance, '
                    save_design'):
123                 print("DEBUG: Calling save_design on instance")
124                 success = design_instance.save_design(popup_summary)
125 
126                 if success:
127                     QMessageBox.information(self, 'Success', 'Design
                            report generated successfully!')
128                     dialog_window.accept()
129                 else:
130                     QMessageBox.warning(self, 'Error', 'Failed to
                            generate design report. Please check the inputs.
                            ')
131             else:
132                 QMessageBox.warning(self, 'Error', 'Could not access
                        design instance for report generation.')
133 
134         except Exception as e:
135             QMessageBox.critical(self, 'Error', f'Error generating
                    report: {str(e)}')
136             print(f"Report generation error: {str(e)}")
137             import traceback
138             traceback.print_exc()
139 
140     def getPopUpInputs(self):
141         """Enhanced method to collect all popup inputs"""
142         input_summary = {}
143         input_summary["ProfileSummary"] = {}
144 
145         # Profile information
146         input_summary["ProfileSummary"]["CompanyName"] = str(self.
                new_ui.lineEdit_companyName.text()) if hasattr(self.new_ui,
                'lineEdit_companyName') else "Company Name"
147         input_summary["ProfileSummary"]["CompanyLogo"] = str(self.
                new_ui.lbl_browse.text()) if hasattr(self.new_ui, '
                lbl_browse') else ""
148         input_summary["ProfileSummary"]["Group/TeamName"] = str(self.
                new_ui.lineEdit_groupName.text()) if hasattr(self.new_ui, '
```

```
            lineEdit_groupName') else "Design Team"
149         input_summary["ProfileSummary"]["Designer"] = str(self.new_ui.
                lineEdit_designer.text()) if hasattr(self.new_ui, '
                lineEdit_designer') else "Designer"
150
151         # Project information - **CRITICAL**: These are required by
                reportGenerator
152         input_summary["ProjectTitle"] = str(self.new_ui.
                lineEdit_projectTitle.text()) if hasattr(self.new_ui, '
                lineEdit_projectTitle') else "Welded Butt Joint Design"
153         input_summary["Subtitle"] = str(self.new_ui.lineEdit_subtitle.
                text()) if hasattr(self.new_ui, 'lineEdit_subtitle') else "
                Design Report"
154         input_summary["JobNumber"] = str(self.new_ui.lineEdit_jobNumber
                .text()) if hasattr(self.new_ui, 'lineEdit_jobNumber') else
                "JOB-001"
155         input_summary["Client"] = str(self.new_ui.lineEdit_client.text
                ()) if hasattr(self.new_ui, 'lineEdit_client') else "Client"
156
157         # Additional comments
158         if hasattr(self.new_ui, 'txt_additionalComments'):
159             input_summary["AdditionalComments"] = str(self.new_ui.
                    txt_additionalComments.toPlainText())
160         else:
161             input_summary["AdditionalComments"] = "Design completed
                    successfully"
162
163         # File and folder settings
164         if hasattr(self.new_ui, 'lineEdit_fileName'):
165             input_summary["filename"] = str(self.new_ui.
                    lineEdit_fileName.text())
166         if hasattr(self.new_ui, 'lineEdit_folder'):
167             input_summary["folder"] = str(self.new_ui.lineEdit_folder.
                    text())
168
169         return input_summary
170 %-------------------- end code ---------------
```

4.3.4 Explanation of the Code

The 3D image generation system creates comprehensive visual documentation for the welded butt joint by generating four different viewing angles (3D, front, top, side) using off-screen rendering to avoid popup windows, but only executes if the design calculations were successful, while implementing robust error handling that ensures the necessary directory structure exists even if image generation fails and safely manages the transition between normal display and off-screen rendering modes.

The report generation controller function serves as the critical bridge between the user interface and the actual report generation system by collecting all popup inputs, solving the complex problem of accessing design calculation results through multiple fallback methods (first trying to retrieve existing design instances, then creating new ones and transferring all calculated data), ensuring the design instance has all necessary attributes and calculation results, and finally executing the actual report generation while providing comprehensive user feedback and error handling throughout the entire process.

The input collection system systematically extracts all user-entered information from the report generation popup dialog by safely accessing UI text fields with comprehensive error handling that provides default values when fields are missing or empty, organizing the collected data into a structured dictionary format that includes profile information (company, designer), critical project metadata (title, job number, client), file management settings (filename, folder path), and additional comments, while using defensive programming techniques to prevent crashes from missing UI elements and ensure the report generator always receives complete and valid input data.

4.4 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/
        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 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

Conclusions

5.1 Tasks Accomplished

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

- Developed a custom LaTeX report generator that allows users to select specific sections and components from a PyLaTeX-generated report for tailored PDF output.
- Redesigned Osdag's interface providing an user-centered design and progressive disclosure.
- Developed the design report generation mechanism for the Welded Butt Joint Connections.
- Integrated the report generation part with the main UI of Osdag keeping the rest functionalities undisrupted.

5.2 Skills Developed

All the tasks provided an opportunity to develop and strengthen several technical and professional skills, including:

- **Advanced Python Programming Skills**

    * Complex class design and inheritance patterns.
    * Attribute management and method implementation.
    * Fail-safe mechanisms with default value fallbacks.
- **LaTeX Document Generation & Mathematical Typesetting**
    * Integration of Python variables with LaTeX templates using PyLaTeX.
    * Complex mathematical equation formatting with subscripts/superscripts.
    * Equation alignment and mathematical expression formatting.
    * Greek letter and special character handling.
- **Structural Engineering based Software Development**
    * IS 800:2007 code compliance and design verification algorithms.
    * Complex mathematical equation formatting with subscripts/superscripts.
    * Metal strength calculations, capacity analysis and failure mode assessment
- **Software Integration & Architecture**
    * Seamless integration with existing Osdag framework.
    * Maintaining consistency with other connection modules.
    * Separation of concerns between calculation, validation, and reporting.
    * Reusable component development and scalable architecture patterns.
- **User Interface & Experience Design**
    * Dialog box design and popup interface creation.
    * Dynamic UI element management and state control.
    * User input validation and data collection systems.

Chapter A

Appendix

A.1 Work Reports

Internship Work Report

NAME **Roushan Raj**
PROJECT **Osdag**
INTERNSHIP **FOSSEE Summer Fellowship 2025**

DATE	DAY	TASK	HOURS WORKED
05/15/25	Thursday	Initial setup and familiarization with Osdag Codebase	6
05/16/25	Friday	Analyzed current GUI of Osdag and looked for the scope of improvement in the design.	6
05/17/25	Saturday	Designed wireframes for the new GUI	7
05/18/25	Sunday	Holiday	0
05/19/25	Monday	Implemented UI improvements through PyQt	6
05/20/25	Tuesday	Created tabbed interface and user-centric design	6
05/21/25	Wednesday	Implemented progressive enclosure and labelled parameters	7
05/22/25	Thursday	Developed improved results visualisation components	5
05/23/25	Friday	Implemented advanced search and filter functionalities.	8
05/24/25	Saturday	Completed GUI and finished my presentation.	6
05/25/25	Sunday	Holiday	0
05/26/25	Monday	Installed Osdag on Ubuntu 22.04LTS, configured the development environment.	7
05/27/25	Tuesday	Understood Osdag's working interface and performed initial testing.	6
05/28/25	Wednesday	Understood Osdag's codebase and file structure.	6
05/29/25	Thursday	Learnt how report of other modules are generated using the DDCL and the respective functions.	6
05/30/25	Friday	Explored latex docs and other provided resources for proper understanding.	7
05/31/25	Saturday	Made a workflow of my assigned task and got reviewed by the mentors.	7
06/01/25	Sunday	Holiday	0
06/02/25	Monday	Implemented the changes and started working on the report generation part.	7
06/03/25	Tuesday	Implemented functions as required in the DDCL of welded_butt_joint module.	8
06/04/25	Wednesday	Performed other design and calculation checks	7
06/05/25	Thursday	Started developing save_design() function of this module.	6
06/06/25	Friday	Fixed small bugs in the existing functions that were hindering the workflow.	5
06/07/25	Saturday	Fixed circular import and other import issues via help of proper modular coding.	6
06/08/25	Sunday	Holiday	0
06/09/25	Monday	Discussed the update with the mentors and corrected the mistakes.	6
06/10/25	Tuesday	Fixed missing keys issue and delay in global import issues.	6

06/11/25	Wednesday	Implemented other functions required according to the DDCL.	7
06/12/25	Thursday	Created the first draft report by taking a test script and got reviewed.	6
06/13/25	Friday	Fixed the Latex formatting issues.	7
06/14/25	Saturday	Fixed the mathematical equations and greek symbols formatting.	7
06/15/25	Sunday	Holiday	0
06/16/25	Monday	Fixed the tabular representation and the overflow problem.	6
06/17/25	Tuesday	Created the second draft report and got reviewed.	5
06/18/25	Wednesday	Added some missing design and detailing checks.	6
06/19/25	Thursday	Certain report content was edited and final report was generated.	7
06/20/25	Friday	Understood how the report generation feature can be linked with the main GUI.	8
06/21/25	Saturday	Started the linking process and modified the ui_template.py	8
06/22/25	Sunday	Holiday	0
06/23/25	Monday	Buttons and labels were added for report generation.	7
06/24/25	Tuesday	ui_summary_popup() file was modified to handle the report generation feature.	7
06/25/25	Wednesday	Glitches were reduced in generation and final ui improvements were made.	7
06/26/25	Thursday	Proper dialog boxes were added for the saving of design reports.	6
06/27/25	Friday	Implemented proper error handling mechanisms.	6
06/28/25	Saturday	Finally reviewed all the ui integration.	6
06/29/25	Sunday	Holiday	0
06/30/25	Monday	Generated the final design report by passing parameters through the main ui.	5
07/01/25	Tuesday	Got reviewed the final performance by the mentors and made the final changes.	7
07/02/25	Wednesday	Commented the code blocks for proper understanding.	5
07/03/25	Thursday	Final touch changes in the code were made with proper print statements..	7
07/04/25	Friday	Added the required repository in my remote and pushed all the codes.	7
07/05/25	Saturday	Progress was dicussed and learnt how to create pull request.	2
07/06/25	Sunday	Holiday	0
07/07/25	Monday	Started documenting the report generation project and its integration with the ui.	5
07/08/25	Tuesday	Created the README.md file for other users along with intructions to test the features.	6
07/09/25	Wednesday	Started framing my pull request	6
07/10/25	Thursday	Created my pull request to the designated branch.	7

07/11/25	Friday	Fixed the pull request issue as informed through the mail.	6
07/12/25	Saturday	Finally made a pull request which got merged successfully.	7
07/13/25	Sunday	Holiday	0
07/14/25	Monday	Final task review and submission.	4
07/15/25	Tuesday	Fellowship conclusion and final wrap-up.	2

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.