



FOSSEE Winter Internship Report

On

Development of New Installer method and Working
on Report Generation of new modules for Osdag

Submitted by

Anuranjani

2nd Year B.Tech Student, Department of Electrical Engineering

Indian Institute of Technology

Jodhpur

Under the Guidance of

Prof. Siddhartha Ghosh

Department of Civil Engineering

Indian Institute of Technology Bombay

Mentors:

Ajmal Babu M S

Parth Karia

Ajinkya Dahale

January 8, 2025

Acknowledgments

- I would like to express my profound gratitude to all those who have supported and guided me throughout this enriching journey. This project has been a significant learning experience, made possible by the collective support of many individuals and institutions.
- My sincere thanks go to the Project staff at the Osdag team - Ajmal Babu M. S., Ajinkya Dahale, and Parth Karia - for their constant support, technical expertise, and willingness to share their knowledge. Their mentorship has been invaluable in helping me overcome challenges and grow professionally.
- I am deeply grateful to Prof. Siddhartha Ghosh, Principal Investigator (PI) of the Osdag project at the Department of Civil Engineering, IIT Bombay, for his invaluable guidance and vision. His leadership has been instrumental in shaping this project and my understanding of the field.
- I extend my heartfelt appreciation to Prof. Kannan M. Moudgalya, FOSSEE PI from the Department of Chemical Engineering, IIT Bombay, for his oversight and direction.
- The FOSSEE managers, Usha Viswanathan and Vineeta Parmar, along with their entire team, have provided exceptional support and created an environment conducive to learning and growth.
- This project would not have been possible without the support of the National Mission on Education through Information and Communication Technology (ICT), Ministry of Education (MoE), Government of India. Their commitment to advancing educational technology has made this opportunity possible.

- I am particularly thankful to my co-interns, whose collaboration and friendship made this journey both enjoyable and intellectually stimulating. The daily interactions, shared challenges, and mutual support have contributed significantly to my professional development.

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 |
| 3 | Internship Task 1: Resolve Import Path Issues and Float Error for Mac UI Template | 17 |
| 3.1 | Task 1: Problem Statement | 17 |
| 3.2 | Task 1: Tasks Done | 17 |
| 3.3 | Task 1: Python Code | 18 |
| 4 | Internship Task 2 Title: Resolve LaTeX Report Issues for Compression Members | 21 |
| 4.1 | Task 2: Problem Statement | 21 |
| 4.2 | Task 2: Tasks Done | 21 |
| 4.3 | Task 2: Python Code | 22 |
| 4.3.1 | Report_functions.py | 22 |
| 4.3.2 | reportGenerator_latex.py | 28 |
| 4.3.3 | Column.py | 29 |
| 4.3.4 | compression.py | 43 |
| 5 | Internship Task 3 Title: Miscellaneous | 54 |

| | | |
|----------|---|-----------|
| 5.1 | Task 3: Problem Statement | 54 |
| 5.2 | Task 3: Tasks Done | 54 |
| 5.3 | Task 3: Python Code | 55 |
| 5.3.1 | For fixing Output Dock Designation and Selected Member Data . | 55 |
| 5.3.2 | For fixing Input and Output Dock Images and Sketch | 56 |
| 6 | Internship Task 4 Title: Documentation of Report Generation and the New Installer Method | 59 |
| 6.1 | Task 3: Problem Statement | 59 |
| 6.2 | Task 3: Tasks Done | 59 |
| 6.3 | Task 3: Documentation | 60 |
| 7 | Conclusions | 61 |
| 7.1 | Tasks Accomplished | 61 |
| 7.2 | Skills Developed | 61 |
| A | Appendix | 63 |
| A.1 | Work Reports | 63 |
| | Bibliography | 65 |

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 | SWAYAMPBABHA | Access 24x7 TV programs | Enable SWAYAMPBABHA 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.

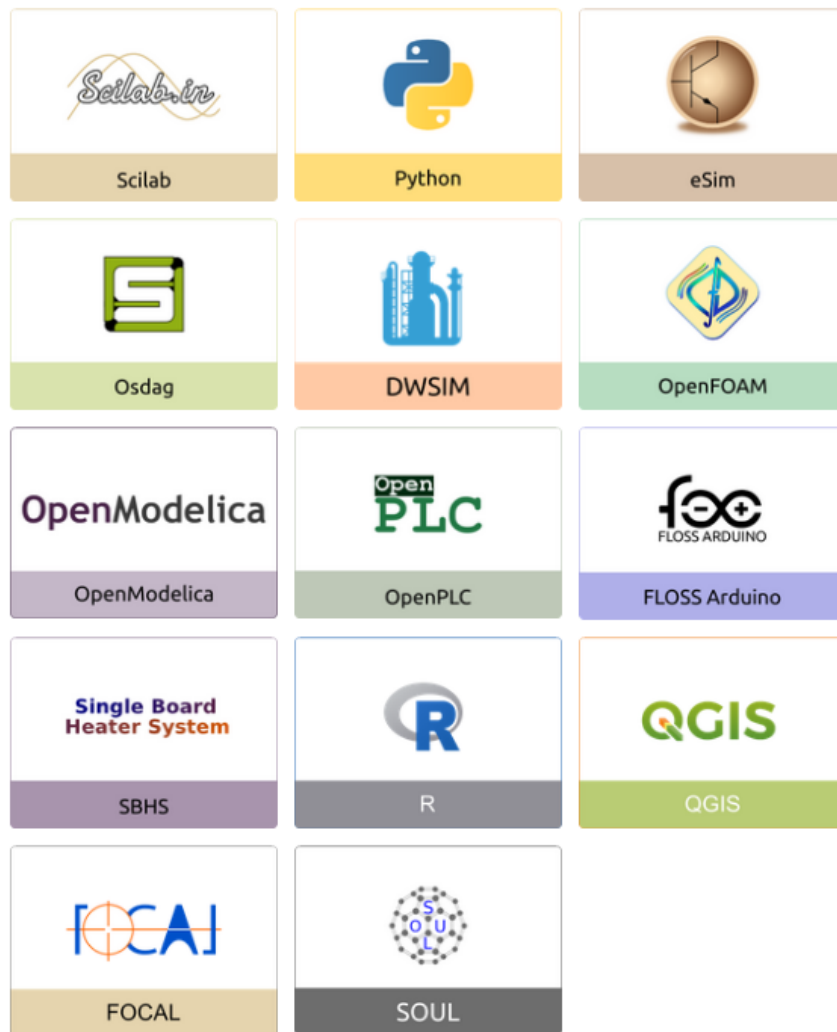


Figure 1.1: FOSSEE Projects and Activities

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.

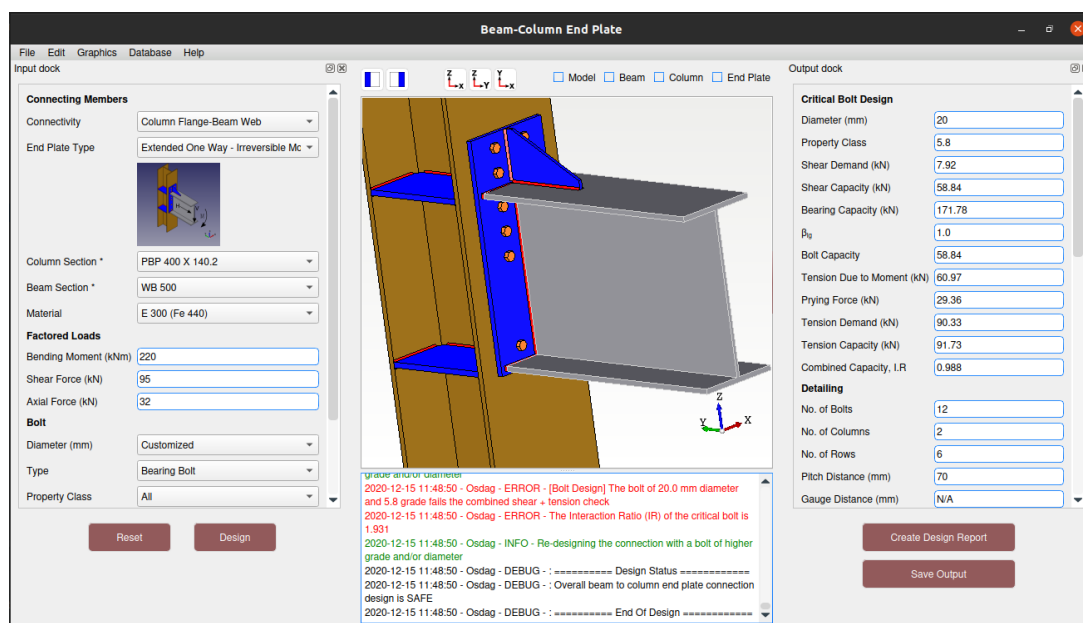


Figure 1.2: Osdag GUI

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

We were provided with a 3D Prism Viewer application that needs to be tested, packaged, and documented. The task is divided into the following key areas:

1. **Unit Testing:** Develop comprehensive unit tests to validate the application's functionality, specifically focusing on the accuracy of surface area and volume calculations. The tests should adhere to Python's best practices for unit testing.
2. **Packaging:** Package the application using Conda, ensuring proper environment specifications and dependency management.
3. **Documentation:** Prepare and submit a detailed report in MS Word or LaTeX format. The report should document the testing methodology adopted and, packaging steps during the task.

2.2 Tasks Done

Below is a summary of the tasks performed as part of the screening process:

1. Developed and executed unit tests to verify the accuracy of surface area and volume calculations, ensuring the application adheres to its functional requirements.

Listing 2.1: Unit Tests File for the Prism Viewer

```

1 import unittest
2 import sqlite3
3 import sys
4 from PyQt5.QtWidgets import QApplication
5 from prism_viewer.main import PrismViewer
6 from prism_viewer.prism_calculator import PrismCalculator
7
8 class TestPrismViewerApp(unittest.TestCase):
9     @classmethod
10    def setUpClass(cls):
11        # Initialize the test database.
12        cls.conn = sqlite3.connect('prisms.db')
13        cls.cursor = cls.conn.cursor()
14
15        # Create the application instance.
16        cls.app = QApplication(sys.argv)
17        cls.viewer = PrismViewer()
18
19    def setUp(self):
20        # Reset the application state before each test.
21        self.viewer.designation_dropdown.setCurrentIndex(0)
22
23    def test_program_initialization(self):
24        self.assertEqual(self.viewer.windowTitle(), "Rectangular
25                          Prism Viewer")
26
27        # Verify essential UI components.
28        self.assertIsNotNone(self.viewer.designation_dropdown)
29        self.assertIsNotNone(self.viewer.surface_area_label)
30        self.assertIsNotNone(self.viewer.volume_label)
31        self.assertIsNotNone(self.viewer.display_button)
32
33        # Check the database connections.
34        self.assertIsNotNone(self.viewer.conn)
35        self.assertIsNotNone(self.viewer.cursor)
36
37        # Check if the viewer has been created or not.

```

```

37         self.assertTrue(self.viewer.isHidden())
38
39     @classmethod
40     def tearDownClass(cls):
41         cls.conn.close()
42
43 class TestPrismCalculator(unittest.TestCase):
44
45     def test_surface_area_calculation(self):
46         test_cases = [
47             # (length, width, height, expected_area = 2 * (length *
48                 width + width * height + height * length))
49             (10, 5, 2, 2 * (10 * 5 + 5 * 2 + 2 * 10)), # Standard
50                 case
51             (1, 1, 1, 6), # Unit
52                 cube
53             (10, 10, 10, 600), # Equal
54                 dimensions
55         ]
56
57         for length, width, height, expected in test_cases:
58             with self.subTest(f"Testing surface area with L={length
59                 }, W={width}, H={height}"):
60                 result = PrismCalculator.surface_area(length, width
61                     , height)
62                 self.assertEqual(result, expected)
63
64     def test_volume_calculation(self):
65         test_cases = [
66             # (length, width, height, expected_volume = length *
67                 width * height)
68             (10, 5, 2, 10*5*2), # Standard case
69             (1, 1, 1, 1), # Unit cube
70             (10, 10, 10, 1000), # Equal dimensions
71         ]
72
73         for length, width, height, expected in test_cases:
74             with self.subTest(f"Testing volume with L={length}, W={
75                 width}, H={height}"):

```

```

68         result = PrismCalculator.volume(length, width,
69             height)
70
71     def test_zero_value_calculation(self):
72         # To check if zero is returned for zero dimensions.
73         # Surface area tests
74         self.assertEqual(PrismCalculator.surface_area(0, 0, 0), 0)
75         self.assertEqual(PrismCalculator.surface_area(5, 0, 0), 0)
76         self.assertEqual(PrismCalculator.surface_area(0, 5, 0), 0)
77         self.assertEqual(PrismCalculator.surface_area(0, 0, 5), 0)
78
79         # Volume tests
80         self.assertEqual(PrismCalculator.volume(0, 0, 0), 0)
81         self.assertEqual(PrismCalculator.volume(0, 5, 2), 0)
82         self.assertEqual(PrismCalculator.volume(5, 0, 2), 0)
83         self.assertEqual(PrismCalculator.volume(5, 2, 0), 0)
84
85     def test_negative_value_calculation(self):
86         # To ensure handling of errors for negative values.
87         negative_cases = [
88             (-10, 5, 2),
89             (10, -5, 2),
90             (10, 5, -2),
91             (-1, -1, -1)
92         ]
93
94         for length, width, height in negative_cases:
95             with self.subTest(f"Testing negative values L={length},
96                 W={width}, H={height}"):
97                 with self.assertRaises(ValueError):
98                     PrismCalculator.surface_area(length, width,
99                         height)
100                 with self.assertRaises(ValueError):
101                     PrismCalculator.volume(length, width, height)
102
103 if __name__ == '__main__':
104     unittest.main()

```

2. Packaged the Prism Viewer application using Conda, creating a well-defined environment with all dependencies managed effectively.

Listing 2.2: Conda-Recipe for the application

```
1 {% set name = "prism_viewer" %}
2 {% set version = "0.1.1" %}
3
4 package:
5     name: {{ name|lower }}
6     version: {{ version }}
7
8 source:
9     path: ..
10
11 build:
12     number: 0
13     script: "{{ PYTHON }}" -m pip install . -vv"
14     entry_points:
15         - prism_viewer = prism_viewer.main:main
16     skip: true # [py<311]
17
18 requirements:
19     build:
20         - python =3.11
21         - {{ compiler('cxx') }}
22     host:
23         - python =3.11
24         - pip
25         - setuptools =75.3.0
26         - numpy =1.26.4
27         - swig
28         - pythonocc-core =7.8.1
29         - pyqt =5.15.9
30     run:
31         - python =3.11
32         - {{ pin_compatible('numpy') }}
33         - pyqt =5.15.9
34         - pythonocc-core =7.8.1
35         - sqlite
```

```

36     - occt =7.8.1
37     - six =1.16.0
38     - svgwrite
39     - qt =5.15.9
40
41 test:
42     imports:
43     - prism_viewer
44     - OCC
45     - PyQt5
46     requires:
47     - unittest-xml-reporting
48     commands:
49     - python -m unittest discover -s prism_viewer/tests/
50     source_files:
51     - prism_viewer/tests/
52
53 about:
54     home: "https://github.com/anuranjani23/fossee-3D-rectangular-
55           prism-viewer.git"
56     license: MIT
57     license_family: MIT
58     license_file: LICENSE
59     summary: "A PyQt5 and PythonOCC-based 3D rectangular prism viewer
60           application"
61     description: |
62     A 3D viewer application for rectangular prisms built using
63     PyQt5 and PythonOCC.
64     Features include surface area and volume calculations,
65     interactive 3D visualization,
66     and SQLite database storage.
67     doc_url: https://github.com/anuranjani23/fossee-3D-rectangular-
68           prism-viewer/blob/main/README.md
69     dev_url: https://github.com/anuranjani23/fossee-3D-rectangular-
70           prism-viewer
71
72 extra:
73     recipe-maintainers:
74     - anuranjani23

```



```
69 platforms:
70   - linux
71   - osx
72   - win-64
```

Listing 2.3: Enviornment YML file

```
1 name: prism_viewer_env
2 channels:
3   - conda-forge
4 dependencies:
5   - python=3.11
6   - pyqt=5.15.9
7   - numpy=1.26.4
8   - pythonocc-core=7.8.1
9   - swig
10  - sqlite
11  - occt=7.8.1
12  - six=1.16.0
13  - svgwrite
14  - qt=5.15.9
15  - pip
```

3. Documented the entire process, including testing approaches, packaging methods, and the rationale behind critical decisions, in a structured report prepared using LaTeX. The report can be found [here](#).
4. Demonstrated proficiency in Python programming, focusing on Object-Oriented Programming (OOP) principles, unit testing, and industry-standard packaging practices with Conda and PIP building and packaging.

Chapter 3

Internship Task 1: Resolve Import Path Issues and Float Error for Mac UI Template

3.1 Task 1: Problem Statement

The task involved resolving critical issues in the Mac UI template concerning float handling and import paths. The primary challenges involved debugging and fixing float-related errors that were affecting the UI rendering, along with restructuring the import system.

3.2 Task 1: Tasks Done

My responsibilities included identifying and resolving float computation errors, removing redundant imports that were cluttering the codebase, and standardizing the import paths to use relative references for better maintainability. This optimization task aimed to enhance the template's reliability and maintain consistent coding standards across the UI framework.

3.3 Task 1: Python Code

This section presents the changes made to the Python script for the UI template of Mac.

The figures below illustrate these changes:

```
src/osdag/gui/ui_template_for_mac.py
73 85 from ..design_type.connection.column_end_plate import ColumnEndPlate
74 86 from ..design_type.connection.column_cover_plate_weld import ColumnCoverPlateWeld
75 87 from ..design_type.connection.base_plate_connection import BasePlateConnection
@@ -78,7 +90,6 @@
78 90 import logging
79 91 import subprocess
80 92 from ..get_DPI_scale import scale
81 - from ..cad.cad3dconnection import cadconnection
82 93 from OCC.Display.backend import load_backend, get_qt_modules
83 94 from ..osdagMainSettings import backend_name
84 95 used_backend = load_backend(backend_name())
@@ -121,7 +132,7 @@ def __init__(self, main, folder, parent=None):
121 132 resolution = QtWidgets.QDesktopWidget().screenGeometry()
122 133 width = resolution.width()
123 134 height = resolution.height()
124 - self.resize(width * (0.75), height * (0.7))
135 + self.resize(int(width * 0.75), int(height * 0.7))
125 136 self.ui = Window()
126 137 self.ui.setupUi(self, main, folder)
127 138 # self.showMaximized()
@@ -146,21 +157,22 @@ def resize_dockComponents(self):
146 157 # Input Dock
147 158 width = self.ui.inputDock.width()
148 159 self.ui.inputDock.resize(width, self.height())
149 - self.ui.in_widget.resize(width, posi)
160 + self.ui.in_widget.resize(int(width), int(posi))
150 161
151 - self.ui.btn_Reset.move((width / 2) - 110, posi + 8)
152 - self.ui.btn_Design.move((width / 2) + 17, posi + 8)
162 +
163 + self.ui.btn_Reset.move(int((width / 2)) - 110, int(posi + 8))
164 + self.ui.btn_Design.move(int((width / 2)) + 17, int(posi + 8))
153 165 # self.ui.btn_Design.move(,posi+10)
154 166
155 167 # Output Dock
156 168 width = self.ui.outputDock.width()
157 169 self.ui.outputDock.resize(width, self.height())
158 - self.ui.out_widget.resize(width, posi)
159 - self.ui.btn_CreateDesign.move((width / 2) - (186 / 2), posi + 8)
160 - self.ui.save_outputDock.move((width / 2) - (186 / 2), posi + 52)
170 + self.ui.out_widget.resize(int(width), int(posi))
171 + self.ui.btn_CreateDesign.move(int((width / 2) - (186 / 2)), int(posi + 8))
172 + self.ui.save_outputDock.move(int((width / 2) - (186 / 2)), int(posi + 52))
161 173
162 174 # Designed model
163 - self.ui.splitter.setSizes([0.85 * posi, 0.15 * posi])
175 + self.ui.splitter.setSizes([int(0.85 * posi), int(0.15 * posi)])
164 176 self.ui.modelTab.setFocus()
165 177 self.ui.display.FitAll()
166 178
@@ -235,7 +247,7 @@ def open_summary_popup(self, main):
```

```

src/osdag/gui/ui_template_for_mac.py
@@ -235,7 +247,7 @@ def open_summary_popup(self, main):
235 247
236 248     self.new_window = QtWidgets.QDialog(self)
237 249     self.new_ui = Ui_Dialog1(main.design_status, loggermsg=self.textEdit.toPlainText())
238 -     self.new_ui.setupUi(self.new_window, main)
250 +     self.new_ui.setupUi(self.new_window, main, main)
239 251     self.new_ui.btn_browse.clicked.connect(lambda: self.getLogoFilePath(self.new_window, self.new_ui.lbl_browse))
240 252     self.new_ui.btn_saveProfile.clicked.connect(lambda: self.saveUserProfile(self.new_window))
241 253     self.new_ui.btn_useProfile.clicked.connect(lambda: self.useUserProfile(self.new_window))

@@ -733,8 +745,9 @@ def setupUi(self, MainWindow, main, folder):
733 745     maxi_width += 82
734 746     print('maxiwidth',maxi_width)
735 747     maxi_width = max(maxi_width, scale*350) # In case there is no widget
736 -     self.inputDock.setFixedWidth(maxi_width)
737 -     self.in_widget.setFixedWidth( maxi_width)
748 +     self.inputDock.setFixedWidth(int(maxi_width))
749 +
750 +     self.in_widget.setFixedWidth(int(maxi_width))
738 751     for option in option_list:
739 752         key = self.dockWidgetContents.findChild(QtWidgets.QWidget, option[0])
740 753

@@ -812,7 +825,7 @@ def setupUi(self, MainWindow, main, folder):
812 825         self.on_change_connect(key_changed, updated_list, data, main)
813 826
814 827     self.btn_Reset = QtWidgets.QPushButton(self.dockWidgetContents)
815 -     self.btn_Reset.setGeometry(QtCore.QRect((maxi_width/2)-110, 650, 100, 35))
828 +     self.btn_Reset.setGeometry(QtCore.QRect(int((maxi_width / 2) - 110), 650, 100, 35))
816 829     font = QtGui.QFont()
817 830     font.setPointSize(10)
818 831     font.setBold(True)

@@ -822,7 +835,7 @@ def setupUi(self, MainWindow, main, folder):
822 835     self.btn_Reset.setObjectName("btn_Reset")
823 836
824 837     self.btn_Design = QtWidgets.QPushButton(self.dockWidgetContents)
825 -     self.btn_Design.setGeometry(QtCore.QRect((maxi_width/2)+10, 650, 100, 35))
838 +     self.btn_Design.setGeometry(QtCore.QRect(int((maxi_width / 2) + 10), 650, 100, 35))
826 839     font = QtGui.QFont()
827 840     font.setPointSize(10)
828 841     font.setBold(True)

@@ -995,8 +1008,8 @@ def setupUi(self, MainWindow, main, folder):
995 1008     out_scroll.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)
996 1009     out_scroll.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAsNeeded)
997 1010
998 -     self.outputDock.setFixedWidth(maxi_width)
999 -     self.out_widget.setFixedWidth(maxi_width)
1011 +     self.outputDock.setFixedWidth(int(maxi_width))
1012 +     self.out_widget.setFixedWidth(int(maxi_width))
1000 1013     self.outputDock.setSizePolicy(QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Maximum,QtWidgets.QSizePolicy.Maximum))
1001 1014     self.out_widaet.setSizePolicv(QtWidaets.QSizePolicyv(QtWidaets.QSizePolicyv.Maximum,QtWidaets.QSizePolicyv.Maximum))

```

```

src/osdag/gui/ui_template_for_mac.py
995 1008     out_scroll.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)
996 1009     out_scroll.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAsNeeded)
997 1010
998 -       self.outputDock.setFixedWidth(maxi_width)
999 -       self.out_widget.setFixedWidth(maxi_width)
1011 +      self.outputDock.setFixedWidth(int(maxi_width))
1012 +      self.out_widget.setFixedWidth(int(maxi_width))
1000 1013     self.outputDock.setSizePolicy(QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Maximum,QtWidgets.QSizePolicy.Maximum))
1001 1014     self.out_widget.setSizePolicy(QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Maximum,QtWidgets.QSizePolicy.Maximum))
1002 1015     # common_button = QtWidgets.QPushButton()

@@ -1052,7 +1065,7 @@ def setupUi(self, MainWindow, main, folder):
1052 1065     self.save_outputDock.setObjectName("save_outputDock")
1053 1066     self.save_outputDock.setText("Save Output")
1054 1067     self.save_outputDock.clicked.connect(self.save_output_to_csv(main))
1055 -     # self.btn_CreateDesign.clicked.connect(self.createDesignReport(main))
1068 +     #self.btn_CreateDesign.clicked.connect(self.createDesignReport(main))
1056 1069
1057 1070     #####
1058 1071     # Menu UI

@@ -1407,7 +1420,7 @@ def setupUi(self, MainWindow, main, folder):
1407 1420         out_titles.append(title_name)
1408 1421     self.ui_loaded = True
1409 1422
1410 -     from osdagMainSettings import backend_name
1423 +     from ..osdagMainSettings import backend_name
1411 1424     self.display, _ = self.init_display(backend_str=backend_name(),window=MainWindow)
1412 1425     self.connectivity = None
1413 1426     self.fuse_model = None

@@ -1816,8 +1829,8 @@ def return_class(self,name):
1816 1829         return BeamCoverPlate
1817 1830     elif name == KEY_DISP_BEAMCOVERPLATEWELD:
1818 1831         return BeamCoverPlateWeld
1819 -     elif name == KEY_DISP_BEAMENDPLATE:
1820 -     return BeamEndPlate
1832 +     elif name == KEY_DISP_BB_EP_SPLICE:
1833 +     return BeamBeamEndPlateSplice
1821 1834     elif name == KEY_DISP_COLUMNENDPLATE:
1822 1835         return ColumnEndPlate
1823 1836     elif name == KEY_DISP_BASE_PLATE:

```

Chapter 4

Internship Task 2 Title: Resolve LaTeX Report Issues for Compression Members

4.1 Task 2: Problem Statement

Fixing and completing the Report Generation for both modules of Compression Members, "Columns with known support conditions" and "Struts in Trusses".

4.2 Task 2: Tasks Done

The task primarily focused on debugging the Python source files associated with both modules to ensure the successful generation of the report. This involved addressing errors in the code and ensuring all components worked seamlessly together. The next step was to complete and refine the code responsible for generating the content of the report, including integrating CAD-generated images into the appropriate sections of the document.

To achieve this, several new functions were written, while existing functions were carefully reviewed and fixed to address any issues. Additionally, meticulous attention was given to verifying calculations, ensuring accurate results. The LaTeX formatting of the report was also reviewed and corrected to maintain a cleaner appearance, including proper alignment, spacing, and consistency. The integration of images into the report

was carefully managed to ensure they were placed in the correct order and contextually aligned with the content. With this, addition of the failed design Report was also ensured, i.e. if the CAD generation fails, the Report is able to generate still.

Another critical aspect of the task was fixing the design log. This involved resolving issues with excessive or overflowing logger messages, which were being inappropriately printed in the report. By implementing these fixes, the task aimed to ensure that the report generation process was robust, accurate, and met all requirements.

4.3 Task 2: Python Code

The figures below describe the changes made, and the new functions that were added accordingly:

4.3.1 Report_functions.py

Firstly in the `Report_functions.py`, the functions that were added are as follows: Both the `c1_3_7_2_section_classification_angle_required` function and the `c1_3_7_2_section_classification_angle_provided` function were responsible for the Section Classification check in the "Struts in Trusses" module. Then in the same module, the `c1_7_5_1_2_effective_slenderness_ratio` function was added which is used to print Slenderness Ratio when `load.type` is not "Concentric Load". Then after that, `calculate_buckling_class` was added to calculate the Buckling Class curve for the "Column with known support conditions" module, using this function, both the functions `comp_column_class_section_check_required` and `comp_column_class_section_check_provided` were responsible for the addition of Buckling Class - Compatibility check in the Design Check of the Report of same module.

```

src/osdag/Report_functions.py
128 128     eqn.append(NoEscape(r'\begin{aligned} & \textbf{Slender} \end{aligned}'))
129 129     # eqn.append(NoEscape(r'& [\text{Ref: Table 2, Cl.3.7.2 and 3.7.4, IS 800:2007}] \end{aligned}'))
130 130     return eqn
131 +
132 +
133 + def cl_3_7_2_section_classification_angle_required(ratio_type, class_of_section=None):
134 +     """
135 +     Provide the required conditions for angle section classification based on IS 800:2007, Cl.3.7.2.
136 +
137 +     Args:
138 +     ratio_type: Type of ratio to be calculated ('b/t', 'd/t', 'b+d/t')
139 +     class_of_section: Expected classification ('Plastic', 'Compact', 'Semi-Compact', 'Slender')
140 +     epsilon: Material constant (float)
141 +
142 +     Returns:
143 +     A LaTeX equation showing the required classification conditions.
144 +     """
145 +     eqn = Math(inline=True)
146 +
147 +     if class_of_section in ["Plastic", "Compact"]:
148 +         eqn.append(NoEscape(r'\begin{aligned} \text{For ' + class_of_section + r' Section:} \\\'))
149 +         eqn.append(NoEscape(r'\text{No Specific Ratio Limit} \end{aligned}'))
150 +
151 +     elif class_of_section == "Semi-Compact":
152 +         if ratio_type == 'b/t':
153 +             eqn.append(NoEscape(r'\begin{aligned} \\\'))
154 +             eqn.append(NoEscape(r'\frac{b}{t} \leq 15.7\varepsilon'))
155 +             eqn.append(NoEscape(r'\end{aligned}'))
156 +         elif ratio_type == 'd/t':
157 +             eqn.append(NoEscape(r'\begin{aligned} \\\'))
158 +             eqn.append(NoEscape(r'\frac{d}{t} \leq 15.7\varepsilon'))
159 +             eqn.append(NoEscape(r'\end{aligned}'))
160 +         elif ratio_type == '(b+d)/t':
161 +             eqn.append(NoEscape(r'\begin{aligned} \\\'))
162 +             eqn.append(NoEscape(r'\frac{b+d}{t} \leq 25\varepsilon'))
163 +             eqn.append(NoEscape(r'\end{aligned}'))
164 +
165 +     else:
166 +         raise ValueError("Invalid section classification. Choose from 'Plastic', 'Compact', 'Semi-Compact'.")
167 +
168 +     return eqn
169 +
170 + def cl_3_7_2_section_classification_angle_provided(b, d, t, ratio_value, ratio_type, epsilon, class_of_section=None):
171 +     """
172 +     Provide the numerical values for angle section classification based on IS 800:2007, Cl.3.7.2.
173 +
174 +     Args:
175 +     b: Width of the angle leg (float)
176 +     d: Depth of the angle (float)
177 +     t: Thickness of the leg (float)
178 +     ratio_type: Type of ratio to be calculated ('b/t', 'd/t', 'b+d/t')
179 +     ratio_value: The pre-calculated ratio value (float)

```

Figure 4.1: From Report_functions.py


```

src/osdag/Report_functions.py
176 +     d: Depth of the angle (float)
177 +     t: Thickness of the leg (float)
178 +     ratio_type: Type of ratio to be calculated ('b/t', 'd/t', 'b+d/t')
179 +     ratio_value: The pre-calculated ratio value (float)
180 +     class_of_section: Expected classification ('Plastic', 'Compact', 'Semi-Compact', 'Slender')
181 +     epsilon: Material constant (float)
182 +
183 +     Returns:
184 +         A LaTeX equation showing the numerical values and classification.
185 +     """
186 +     eqn = Math(inline=True)
187 +
188 +     if ratio_type == 'b/t':
189 +         eqn.append(NoEscape(r'\begin{aligned}'))
190 +         eqn.append(NoEscape(r'\frac{b}{t} = \frac{' + str(b) + '}' + str(t) + '} = ' + str(ratio_value) + r' \leq 15.7\varepsilon \\\'))
191 +         eqn.append(NoEscape(r'& \textbf{' + class_of_section + r'} \end{aligned}'))
192 +
193 +     elif ratio_type == 'd/t':
194 +         eqn.append(NoEscape(r'\begin{aligned}'))
195 +         eqn.append(NoEscape(r'\frac{d}{t} = \frac{' + str(d) + '}' + str(t) + '} = ' + str(ratio_value) + r' \leq 15.7\varepsilon \\\'))
196 +         eqn.append(NoEscape(r'& \textbf{' + class_of_section + r'} \end{aligned}'))
197 +
198 +     elif ratio_type == '(b+d)/t':
199 +         eqn.append(NoEscape(r'\begin{aligned}'))
200 +         eqn.append(NoEscape(r'\frac{b+d}{t} = \frac{' + str(b+d) + '}' + str(t) + '} = ' + str(ratio_value) + r' \leq 25\varepsilon \\\'))
201 +         eqn.append(NoEscape(r'& \textbf{' + class_of_section + r'} \end{aligned}'))
202 +
203 +     else:
204 +         raise ValueError("Invalid ratio type. Choose from 'b/t', 'd/t', '(b+d)/t'.")
205 +
206 +     return eqn
207 +
208 +
131 209 def cl_5_4_1_table_4_5_gamma_value(v, t):
132 210     """
133 211     Calculate gamma value
@@ -448,6 +526,40 @@ def cl_7_1_2_effective_slenderness_ratio(K, L, r, slender):
448 526     slender_eqn.append(NoEscape(r'& [\text{Ref. IS 800:2007, Cl.7.1.2}] \end{aligned}'))
449 527     return slender_eqn
450 528
529 +def cl_7_5_1_2_effective_slenderness_ratio(k1, k2, k3, lmb_v, lmb_phi, slender):
530 +    """
531 +    Calculate effective slenderness ratio based on given parameters.
532 +
533 +    Args:
534 +        k1: Constant k1 (float)
535 +        k2: Constant k2 (float)
536 +        k3: Constant k3 (float)
537 +        lmb_v: Slenderness parameter  $\lambda_v$  (float)
538 +        lmb_phi: Slenderness parameter  $\lambda_\phi$  (float)
539 +        slender: Effective slenderness ratio  $\lambda_e$  (float)
540 +

```

Figure 4.2: From Report_functions.py

```

src/osdag/Report_functions.py
537 +     lmb_v: Slenderness parameter  $\lambda_v$  (float)
538 +     lmb_phi: Slenderness parameter  $\lambda_\phi$  (float)
539 +     slender: Effective slenderness ratio  $\lambda_e$  (float)
540 +
541 +     Returns:
542 +         LaTeX representation of the effective slenderness ratio calculation.
543 +
544 +     Note:
545 +         Reference:
546 +         IS 800:2007, Cl.7.5.1.2
547 +         """
548 +         k1 = str(k1)
549 +         k2 = str(k2)
550 +         k3 = str(k3)
551 +         lmb_v = str(lmb_v)
552 +         lmb_phi = str(lmb_phi)
553 +         slender = str(slender)
554 +
555 +         slender_eqn = Math(inline=True)
556 +         slender_eqn.append(NoEscape(r'\begin{aligned} \lambda_e &= \sqrt{k_1 + k_2 \cdot \lambda_v^2 + k_3 \cdot \lambda_\phi^2} \\'))
557 +         slender_eqn.append(NoEscape(r'&= \sqrt{' + k1 + r' + ' + k2 + r' \cdot ' + lmb_v + r'^2 + ' + k3 + r' \cdot ' + lmb_phi + r'^2} \\'))
558 +         slender_eqn.append(NoEscape(r'&= ' + slender + r' \\'))
559 +         slender_eqn.append(NoEscape(r'& [\text{Ref. IS 800:2007, Cl.7.5.1.2}] \end{aligned}'))
560 +
561 +     return slender_eqn
562 +
451 563
452 564 def cl_8_2_moment_capacity_member(Pmc, Mdc, M_c):
453 565     """
454 566     @@ -1325,10 +1437,9 @@ def cl_8_7_1_5_buckling_curve(sub = 'c'):
1325 1437
1326 1438     """
1327 1439
1328 -     sub = str(sub)
1329 1440 +     sub = str(sub).upper()
1330 1441     slender_eqn = Math(inline=True)
1331 -     slender_eqn.append(NoEscape(r'\begin{aligned} &= c \\'))
1332 1442 +     slender_eqn.append(NoEscape(r'\begin{aligned} &= ' + sub + r' \\'))
1333 1443     slender_eqn.append(NoEscape(r'& [\text{Ref. IS 800:2007, Cl.8.7.3.1}] \end{aligned}'))
1334 1444     return slender_eqn
1335 1445
4066 4177     return display_eqn
4067 4178
4068 4179
4069 -def get_pass_fail(required, provided, relation='', M1 = '', M2 = ''):
4180 +def get_pass_fail(required, provided, relation='', M1=''):
4070 4181     if provided == 0 or required == 'N/A' or provided == 'N/A' or required == 0:
4071 4182         return ''
4072 4183     else:

```

Figure 4.3: From Report_functions.py

```

src/osdag/Report_functions.py
4071 4182         return ''
4072 4183     else:
@@ -8778,65 +8889,122 @@ def prov_moment_load_bp(moment_input, min_mc, app_moment_load, moment_capacity,
8778 8889         return app_moment_load_eqn
8779 8890
8780 8891
8781 -
8782 -def comp_column_class_section_check_required( bucklingclass , h , bf ):
8892 +def calculate_buckling_class(h, bf, tf, axis):
8893 +    """
8894 +    Determines the buckling class using IS 800:2007 checks.
8895 +    axis: Can be "ZZ" or "YY"
8896 +    """
8897 +    h_bf_ratio = h / bf
8898 +    tf_limit = tf
8899 +
8900 +    if axis == "ZZ":
8901 +        if h_bf_ratio <= 1.2 and tf_limit > 100:
8902 +            return "D"
8903 +        elif h_bf_ratio <= 1.2 and tf_limit <= 100:
8904 +            return "C"
8905 +        elif h_bf_ratio > 1.2 and tf_limit <= 40:
8906 +            return "B"
8907 +        elif h_bf_ratio > 1.2 and 40 < tf_limit <= 100:
8908 +            return "C"
8909 +
8910 +    elif axis == "YY":
8911 +        if h_bf_ratio > 1.2 and tf_limit <= 40:
8912 +            return "A"
8913 +        elif h_bf_ratio > 1.2 and 40 < tf_limit <= 100:
8914 +            return "B"
8915 +        elif h_bf_ratio <= 1.2 and tf_limit <= 100:
8916 +            return "B"
8917 +        elif h_bf_ratio <= 1.2 and tf_limit > 100:
8918 +            return "D"
8919 +    else:
8920 +        return "Invalid Axis"
8921 +def comp_column_class_section_check_required(h, bf, tf, axis):
8783 8922     """
8784 8923     Args:
8785 -         h:Depth of section(mm) (float)
8786 -         bf: Breadth of section(mm) (float)
8787 -         bucklingclass: buckling class (float)
8924 +         h: Depth of section (mm) (float)
8925 +         bf: Breadth of section (mm) (float)
8926 +         tf: Thickness of flange (mm) (float)
8927 +         axis: Axis for buckling class ("YY" or "ZZ") (str)
8788 8928     Returns:
8789 -         bucklingclass_eq
8790 -     Note:
8791 -         Reference: IS 800 Cl.7.1.2.2
8792 -         @author: Duttvik Joshi

```

Figure 4.4: From Report_functions.py

```

src/osdag/Report_functions.py
8790 - Note:
8791 - Reference: IS 800 Cl.7.1.2.2
8792 - @author:Rutvik Joshi
8929 + bucklingclass_eq: LaTeX formatted buckling class equation (Math object)
8793 8930 """
8931 + bucklingclass_eq = Math(inline=True)
8932 +
8933 + # Calculate buckling class
8934 + calculated_buckling_class = calculate_buckling_class(h, bf, tf, axis)
8935 +
8794 8936 bucklingclass_eq=Math(inline=True)
8795 - if bucklingclass==0.34:
8796 - bucklingclass_eq.append(NoEscape(r'\begin{aligned} \frac{h}{b}\text{\textit{f}}>1.2\text{\textit{f}}'))
8937 + if calculated_buckling_class=="A":
8938 + bucklingclass_eq.append(NoEscape(r'\begin{aligned} \frac{h}{b}\text{\textit{f}}>1.2\text{\textit{f}}'))
8797 8939 bucklingclass_eq.append(NoEscape(r' t\text{\textit{f}}<=40\text{\textit{f}}'))
8798 - bucklingclass_eq.append(NoEscape(r' \end{aligned}'))
8799 - elif bucklingclass==0.49:
8940 + bucklingclass_eq.append(NoEscape(r' \end{aligned}'))
8941 + elif calculated_buckling_class=="B":
8800 8942 if h/bf>1.2:
8801 - bucklingclass_eq.append(NoEscape(r'\begin{aligned} \frac{h}{b}\text{\textit{f}}>1.2\text{\textit{f}}'))
8802 - bucklingclass_eq.append(NoEscape(r'40 <= t\text{\textit{f}} <= 100'))
8803 - bucklingclass_eq.append(NoEscape(r' \end{aligned}'))
8943 + if axis=="YY":
8944 + bucklingclass_eq.append(NoEscape(r'\begin{aligned} \frac{h}{b}\text{\textit{f}}>1.2\text{\textit{f}}'))
8945 + bucklingclass_eq.append(NoEscape(r'40 < t\text{\textit{f}} <= 100'))
8946 + bucklingclass_eq.append(NoEscape(r' \end{aligned}'))
8947 + else:
8948 + bucklingclass_eq.append(NoEscape(r'\begin{aligned} \frac{h}{b}\text{\textit{f}}>1.2\text{\textit{f}}'))
8949 + bucklingclass_eq.append(NoEscape(r' t\text{\textit{f}}<=40\text{\textit{f}}'))
8950 + bucklingclass_eq.append(NoEscape(r' \end{aligned}'))
8804 8951 else:
8805 - bucklingclass_eq.append(NoEscape(r'\begin{aligned} \frac{h}{b}\text{\textit{f}}<=1.2\text{\textit{f}}'))
8952 + bucklingclass_eq.append(NoEscape(r'\begin{aligned} \frac{h}{b}\text{\textit{f}}<=1.2\text{\textit{f}}'))
8806 8953 bucklingclass_eq.append(NoEscape(r' t\text{\textit{f}} <= 100'))
8807 - bucklingclass_eq.append(NoEscape(r' \end{aligned}'))
8808 - elif bucklingclass==0.76:
8809 - bucklingclass_eq.append(NoEscape(r'\begin{aligned} \frac{h}{b}\text{\textit{f}}<=1.2\text{\textit{f}}'))
8810 - bucklingclass_eq.append(NoEscape(r' t\text{\textit{f}} > 100'))
8811 - bucklingclass_eq.append(NoEscape(r' \end{aligned}'))
8954 + bucklingclass_eq.append(NoEscape(r' \end{aligned}'))
8955 + elif calculated_buckling_class=="C":
8956 + if h/bf>1.2:
8957 + bucklingclass_eq.append(NoEscape(r'\begin{aligned} \frac{h}{b}\text{\textit{f}}>1.2\text{\textit{f}}'))
8958 + bucklingclass_eq.append(NoEscape(r'40 < t\text{\textit{f}} <= 100'))
8959 + bucklingclass_eq.append(NoEscape(r' \end{aligned}'))
8960 + else:
8961 + bucklingclass_eq.append(NoEscape(r'\begin{aligned} \frac{h}{b}\text{\textit{f}}<=1.2\text{\textit{f}}'))
8962 + bucklingclass_eq.append(NoEscape(r' t\text{\textit{f}} <= 100'))
8963 + bucklingclass_eq.append(NoEscape(r' \end{aligned}'))
8964 + elif calculated_buckling_class=="D":
8965 + if axis=="YY":

```

Figure 4.5: From Report_functions.py


```

▼ src/osdag/design_report/reportGenerator_latex.py
@@ -232,6 +232,19 @@ def save_latex(self, uiObj, Design_Check, reportssummary, filename, rel_path, Dis
232 232         table.end_table_header()
233 233         table.add_hline()
234 234         count = count + 1
235 +         elif check[0] == 'NewTable':
236 +             if count >= 1:
237 +                 # doc.append(NewPage())
238 +                 doc.append(pyl.Command('Needspace', arguments=NoEscape(r'\baselineskip')))
239 +
240 +                 with doc.create(Subsection(check[1])):
241 +                     with doc.create(LongTable(check[2], row_height=1.2)) as table:
242 +                         table.add_hline()
243 +                         table.add_row(('Axes', 'Buckling Class', 'Imperfection Factor', ''), color='OsdagGreen')
244 +                         table.add_hline()
245 +                         table.end_table_header()
246 +                         table.add_hline()
247 +                         count = count + 1
235 248         elif check[0] == "Selected":
236 249             if count >=1:
237 250                 # doc.append(NewPage())
@@ -471,4 +484,4 @@ def save_latex(self, uiObj, Design_Check, reportssummary, filename, rel_path, Dis
471 484
472 485 def color_cell(cellcolor, celltext):
473 486     string = NoEscape(r'\cellcolor{' + cellcolor + r'}{' + celltext + r}')
474 -     return string
487 +     return string

```

Figure 4.7: From reportGenerator_latex.py

4.3.3 Column.py

In the Column.py, section_classification and save_design functions were modified and common_result function was added, a failure dictionary was added as well to fall back to for report generation when design_status fails. The last 4 images were integrated as well.

```

src/osdag/design_type/compression_member/Column.py
@@ -573,6 +573,18 @@ def func_for_validation(self, design_dictionary):
573 573         if flag:
574 574             print(f"\n design_dictionary{design_dictionary}")
575 575             self.set_input_values(self, design_dictionary)
576 +             if self.design_status ==False and len(self.failed_design_dict)>0:
577 +                 logger.error(
578 +                     "Design Failed, Check Design Report"
579 +                 )
580 +                 return # ['Design Failed, Check Design Report'] @TODO
581 +             elif self.design_status:
582 +                 pass
583 +             else:
584 +                 logger.error(
585 +                     "Design Failed. Slender Sections Selected"
586 +                 )
587 +                 return # ['Design Failed. Slender Sections Selected']
576 588         else:
577 589             return all_errors
578 590
@@ -677,7 +689,7 @@ def set_input_values(self, design_dictionary):
677 689         # initialize the design status
678 690         self.design_status_list = []
679 691         self.design_status = False
680 -
692 +         self.failed_design_dict = {}
681 693         flag = self.section_classification(self)
682 694         print(flag)
683 695         if flag:
@@ -775,11 +787,8 @@ def section_classification(self):

```

Figure 4.8: From Column.py

```

v src/osdag/design_type/compression_member/Column.py
@@ -21,7 +21,7 @@
21 21 from ..member import Member
22 22 from ...Report_functions import *
23 23 from ...design_report.reportGenerator_latex import CreateLatex
24 -
24 +from pylatex.utils import NoEscape
25 25
26 26 class ColumnDesign(Member):
27 27
@@ -731,19 +731,28 @@ def section_classification(self):
731 731     self.web_class = IS800_2007.Table2_iii((self.section_property.depth - (2 * self.section_property.flange_thickness)),
732 732                                             self.section_property.web_thickness, self.material_property.fy,
733 733                                             classification_type='Axial compression')
734 +
735 +     web_ratio = (self.section_property.depth - 2 * (
736 +                 self.section_property.flange_thickness + self.section_property.root_radius)) / self.section_property.web_thickness
737 +     flange_ratio = self.section_property.flange_width / 2 / self.section_property.flange_thickness
738 737
739 738     elif (self.sec_profile == VALUES_SEC_PROFILE[1]): # RHS and SHS
740 739         self.flange_class = IS800_2007.Table2_iii((self.section_property.depth - (2 * self.section_property.flange_thickness)),
741 740                                             self.section_property.flange_thickness, self.material_property.fy,
742 741                                             classification_type='Axial compression')
743 742         self.web_class = self.flange_class
744 +
745 +     web_ratio = (self.section_property.depth - 2 * (
746 +                 self.section_property.flange_thickness + self.section_property.root_radius)) / self.section_property.web_thickness
747 +     flange_ratio = self.section_property.flange_width / 2 / self.section_property.flange_thickness
748 746
749 747     elif self.sec_profile == VALUES_SEC_PROFILE[2]: # CHS
750 748         self.flange_class = IS800_2007.Table2_x(self.section_property.out_diameter, self.section_property.flange_thickness,
751 749                                             self.material_property.fy, load_type='axial compression')
752 750         self.web_class = self.flange_class #Why?
753 +
754 +     web_ratio = (self.section_property.depth - 2 * (
755 +                 self.section_property.flange_thickness + self.section_property.root_radius)) / self.section_property.web_thickness
756 +     flange_ratio = self.section_property.flange_width / 2 / self.section_property.flange_thickness
757 754         # print(f"self.web_class{self.web_class}")
758 746 -
759 755 +
760 756     if self.flange_class == 'Slender' or self.web_class == 'Slender':
761 757         self.section_class = 'Slender'
762 758     else:

```

Figure 4.9: From Column.py


```

src/osdag/design_type/compression_member/Column.py
@@ -775,11 +787,8 @@ def section_classification(self):
775 787         elif self.flange_class == 'Semi-Compact' and self.web_class == 'Semi-Compact':
776 788             self.section_class = 'Semi-Compact'
777 789
778 -         logger.info("The flange of the trial section ({} is {} and web is {}). The section is {} [Reference: Cl 3.7, IS 800:2007].".
779 -                     format(trial_section, self.flange_class, self.web_class, self.section_class))
780 -
781 790         # 2.2 - Effective length
782 -         self.effective_length_zz = IS800_2007.cl_7_2_2_effective_length_of_prismatic_compression_members(
791 +         self.effective_length_zz = IS800_2007.cl_7_2_2_effective_length_of_prismatic_compression_members(
783 792             self.length_zz,
784 793             end_1=self.end_1_z,
785 794             end_2=self.end_2_z)
@@ -789,7 +798,7 @@ def section_classification(self):
789 798         # self.sec_profile) / self.length_yy # mm
790 799         # print(f"self.effective_length {self.effective_length_yy} ")
791 800
792 -         self.effective_length_yy = IS800_2007.cl_7_2_2_effective_length_of_prismatic_compression_members(
801 +         self.effective_length_yy = IS800_2007.cl_7_2_2_effective_length_of_prismatic_compression_members(
793 802             self.length_yy,
794 803             end_1=self.end_1_y,
795 804             end_2=self.end_2_y)
@@ -810,16 +819,17 @@ def section_classification(self):
810 819         # print("+++++")
811 820
812 821         # 2.3 - Effective slenderness ratio
813 -         self.effective_sr_zz = self.effective_length_zz / self.section_property.rad_of_gy_z
814 -         self.effective_sr_yy = self.effective_length_yy / self.section_property.rad_of_gy_y
822 +         self.effective_sr_zz = self.effective_length_zz / self.section_property.rad_of_gy_z
823 +         self.effective_sr_yy = self.effective_length_yy / self.section_property.rad_of_gy_y
815 824
816 -         limit = IS800_2007.cl_3_8_max_slenderness_ratio(1)
817 -         if self.effective_sr_zz > limit and self.effective_sr_yy > limit:
825 +             limit = IS800_2007.cl_3_8_max_slenderness_ratio(1)
826 +             if self.effective_sr_zz > limit and self.effective_sr_yy > limit:
818 827                 logger.warning("Length provided is beyond the limit allowed. [Reference: Cl 3.8, IS 800:2007]")
819 828                 logger.error("Cannot compute. Given Length does not pass.")
820 829                 local_flag = False
821 -         else:
822 -             logger.info("Length provided is within the limit allowed. [Reference: Cl 3.8, IS 800:2007]")
830 +             #else:
831 +             #     logger.info("Length provided is within the limit allowed. [Reference: Cl 3.8, IS 800:2007]")
832 +
823 833
824 834         # if len(self.allowed_sections) == 0:
825 835         #     logger.warning("Select at-least one type of section in the design preferences tab.")
@@ -828,13 +838,11 @@ def section_classification(self):
828 838         # self.design_status_list.append(self.design_status)
829 839
830 840         #TODO: @danish check this part
831 -         # if self.section_class in self.allowed_sections:
832 -         #     self.input_section_list.append(trial_section)

```

Figure 4.10: From Column.py

```

src/osdag/design_type/compression_member/Column.py
831 + # logger.info("Length provided is within the limit allowed. [Reference: Cl 3.8, IS 800:2007]")
832 +
823 833
824 834 # if len(self.allowed_sections) == 0:
825 835 # logger.warning("Select at-least one type of section in the design preferences tab.")
@@ -828,13 +838,11 @@ def section_classification(self):
828 838 # self.design_status_list.append(self.design_status)
829 839
830 840 #TODO: @danish check this part
831 - # if self.section_class in self.allowed_sections:
832 - # self.input_section_list.append(trial_section)
833 - # self.input_section_classification.update({trial_section: self.section_class})
841 + # if self.section_class in self.allowed_sections:
842 + self.input_section_list.append(trial_section)
843 + self.input_section_classification.update({trial_section: [self.section_class, self.flange_class, self.web_class, flange_ratio, web_ratio]})
834 844 # print(f"self.section_class{self.section_class}")
835 845
836 - self.input_section_list.append(trial_section)
837 - self.input_section_classification.update({trial_section: [self.section_class, self.flange_class, self.web_class, flange_ratio, web_ratio]})
838 846
839 847 return local_flag
840 848
@@ -865,19 +873,33 @@ def design_column(self):
865 873 # self.design_status = False
866 874 # self.design_status_list.append(self.design_status)
867 875 self.epsilon = math.sqrt(250 / self.material_property.fy)
868 - if len(self.input_section_list) > 0:
876 + #if len(self.input_section_list) > 0:
877 +
878 + # initializing lists to store the optimum results based on optimum UR and cost
879 +
880 + # 1- Based on optimum UR
881 + self.optimum_section_ur_results = {}
882 + self.optimum_section_ur = []
869 883
870 - # initializing lists to store the optimum results based on optimum UR and cost
884 + # 2 - Based on optimum cost
885 + self.optimum_section_cost_results = {}
886 + self.optimum_section_cost = []
887 + self.flag = self.section_classification(self)
871 888
872 - # 1- Based on optimum UR
873 - self.optimum_section_ur_results = {}
874 - self.optimum_section_ur = []
889 + print('self.flag:',self.flag)
890 + # reduction of the area based on the connection requirements (input from design preferences)
891 + if self.effective_area_factor < 1.0:
892 + self.effective_area = round(self.effective_area * self.effective_area_factor, 2)
875 893
876 - # 2 - Based on optimum cost
877 - self.optimum_section_cost_results = {}
878 - self.optimum_section_cost = []

```

Figure 4.11: From Column.py

```

src/osdag/design_type/compression_member/Column.py
876 - # z - based on optimum cost
877 - self.optimum_section_cost_results = {}
878 - self.optimum_section_cost = []
894 + #logger.warning("Reducing the effective sectional area as per the definition in the Design Preferences tab.")
895 + #logger.info("The actual effective area is {} mm2 and the reduced effective area is {} mm2 [Reference: Cl. 7.3.2, IS 800:2007]").
896 + # format(round((self.effective_area / self.effective_area_factor), 2), self.effective_area))
897 + #else:
898 + #if self.section_class != 'Slender':
899 + # logger.info("The effective sectional area is taken as 100% of the cross-sectional area [Reference: Cl. 7.3.2, IS 800:2007].")
879 900
880 - i = 1
901 + #print('self.input_section_list:',self.input_section_list)
902 + if self.flag:
881 903     for section in self.input_section_list: # iterating the design over each section to find the most optimum section
882 904
883 905         # fetching the section properties of the selected section
906 922     def design_column(self):
907 923         self.section_property = Column(designation=section, material_grade=self.material)
908 924         self.material_property.connect_to_database_to_get_fy_fu(self.material, max(self.section_property.flange_thickness,
909 925         self.section_property.web_thickness))
910 926
911 927
912 928
913 934     def design_column(self):
914 935         self.list_yy.append(section)
915 936
916 937         # Step 1 - computing the effective sectional area
917 938         self.section_class = self.input_section_classification[section]
918 939         self.section_class = self.input_section_classification[section][0]
919 940
920 941         if self.section_class == 'Slender':
921 942             logger.warning("The trial section ({} is Slender. Computing the Effective Sectional Area as per Sec. 9.7.2, "
922 943             "Fig. 2 (B & C) of The National Building Code of India (NBC), 2016.".format(section))
923 944             #logger.warning("The trial section ({} is Slender. Computing the Effective Sectional Area as per Sec. 9.7.2, "
924 945             "# Fig. 2 (B & C) of The National Building Code of India (NBC), 2016.".format(section))
925 946
926 947             if (self.sec_profile == VALUES_SEC_PROFILE[0]): # Beams and Columns
927 948                 self.effective_area = (2 * ((31.4 * self.epsilon * self.section_property.flange_thickness) *
928 949                 self.section_property.flange_thickness)) + \
929 950                 (2 * ((21 * self.epsilon * self.section_property.web_thickness) * self.section_property.web_thickness))
930 951                 (2 * ((21 * self.epsilon * self.section_property.web_thickness) * self.section_property.web_thickness))
931 952             elif (self.sec_profile == VALUES_SEC_PROFILE[1]):
932 953                 self.effective_area = (2 * 21 * self.epsilon * self.section_property.flange_thickness) * 2
933 954             else:
934 955                 self.effective_area = self.section_property.area # mm2
935 956                 # print(f"self.effective_area{self.effective_area}")
936 957
937 958         # reduction of the area based on the connection requirements (input from design preferences)
938 959         if self.effective_area_factor < 1.0:
939 960             self.effective_area = round(self.effective_area * self.effective_area_factor, 2)
940 961
941 962         logger.warning("Reducing the effective sectional area as per the definition in the Design Preferences tab.")

```

Figure 4.12: From Column.py

```

src/osdag/design_type/compression_member/Column.py
896 907 # initialize lists for updating the results dictionary
897 - list_zz = []
898 - list_yy = []
908 + self.list_zz = []
909 + self.list_yy = []
899 910
900 - list_zz.append(section)
901 - list_yy.append(section)
911 + self.list_zz.append(section)
912 + self.list_yy.append(section)
902 913
903 914 # Step 1 - computing the effective sectional area
904 915 self.section_class = self.input_section_classification[section]
@@ -928,11 +939,11 @@ def design_column(self):
928 939 if self.section_class != 'Slender':
929 940     logger.info("The effective sectional area is taken as 100% of the cross-sectional area [Reference: Cl. 7.3.2, IS 800:2007].")
930 941
931 - list_zz.append(self.section_class)
932 - list_yy.append(self.section_class)
942 + self.list_zz.append(self.section_class)
943 + self.list_yy.append(self.section_class)
933 944
934 - list_zz.append(self.effective_area)
935 - list_yy.append(self.effective_area)
945 + self.list_zz.append(self.effective_area)
946 + self.list_yy.append(self.effective_area)
936 947
937 948 # Step 2 - computing the design compressive stress
938 949
@@ -968,11 +979,11 @@ def design_column(self):
968 979 self.imperfection_factor_zz = IS800_2007.cl_7_1_2_1_imperfection_factor(buckling_class=self.buckling_class_zz)
969 980 self.imperfection_factor_yy = IS800_2007.cl_7_1_2_1_imperfection_factor(buckling_class=self.buckling_class_yy)
970 981
971 - list_zz.append(self.buckling_class_zz)
972 - list_yy.append(self.buckling_class_yy)
982 + self.list_zz.append(self.buckling_class_zz)
983 + self.list_yy.append(self.buckling_class_yy)
973 984
974 - list_zz.append(self.imperfection_factor_zz)
975 - list_yy.append(self.imperfection_factor_yy)
985 + self.list_zz.append(self.imperfection_factor_zz)
986 + self.list_yy.append(self.imperfection_factor_yy)
976 987
977 988 # 2.2 - Effective length
978 989 self.effective_length_zz = IS800_2007.cl_7_2_2_effective_length_of_prismatic_compression_members(self.length_zz ,
@@ -982,45 +993,45 @@ def design_column(self):
982 993     end_1=self.end_1_y,
983 994     end_2=self.end_2_y) # mm
984 995
985 - list_zz.append(self.effective_length_zz)
986 - list_yy.append(self.effective_length_yy)

```

Figure 4.13: From Column.py

```

src/osdag/design_type/compression_member/Column.py
1046 + self.list_zz.append(self.f_cd_1_zz)
self.list_yy.append(self.f_cd_1_yy)
1036 1047
1037 - list_zz.append(self.f_cd_2)
1038 - list_yy.append(self.f_cd_2)
1048 + self.list_zz.append(self.f_cd_2)
1049 + self.list_yy.append(self.f_cd_2)
1039 1050
1040 - list_zz.append(self.f_cd_zz)
1041 - list_yy.append(self.f_cd_yy)
1051 + self.list_zz.append(self.f_cd_zz)
1052 + self.list_yy.append(self.f_cd_yy)
1042 1053
1043 - list_zz.append(self.f_cd)
1044 - list_yy.append(self.f_cd)
1054 + self.list_zz.append(self.f_cd)
1055 + self.list_yy.append(self.f_cd)
1045 1056
1046 1057 # 2.7 - Capacity of the section
1047 1058
1048 1059 self.section_capacity = self.f_cd * self.effective_area # N
1049 1060
1050 - list_zz.append(self.section_capacity)
1051 - list_yy.append(self.section_capacity)
1061 + self.list_zz.append(self.section_capacity)
1062 + self.list_yy.append(self.section_capacity)
1052 1063
1053 1064 # 2.8 - UR
1054 1065 self.ur = round(self.load.axial_force / self.section_capacity, 3)
1055 1066
1056 - list_zz.append(self.ur)
1057 - list_yy.append(self.ur)
1067 + self.list_zz.append(self.ur)
1068 + self.list_yy.append(self.ur)
1058 1069 self.optimum_section_ur.append(self.ur)
1059 1070
1060 1071 # 2.9 - Cost of the section in INR
1061 1072 self.cost = (self.section_property.unit_mass * self.section_property.area * 1e-4) * min(self.length_zz, self.length_yy) *
1062 1073 self.steel_cost_per_kg
1063 1074
1064 - list_zz.append(self.cost)
1065 - list_yy.append(self.cost)
1075 + self.list_zz.append(self.cost)
1076 + self.list_yy.append(self.cost)
1066 1077 self.optimum_section_cost.append(self.cost)
1067 1078 # print(f"list_zz{list_zz}, list_yy{list_yy} ")
1068 1079
@@ -1076,7 +1087,7 @@ def design_column(self):
1076 1087 # 1- Based on optimum UR
1077 1088 self.optimum_section_ur_results[self.ur] = {}
1078 1089
1079 - list_2 = list_zz + list_yy
1090 + list_2 = self.list_zz + self.list_yy
1080 1091 for i in list_1:

```

Figure 4.14: From Column.py

```

src/osdag/design_type/compression_member/Column.py
1107 1118         break
1108 -         else:
1109 -             logger.warning("The section(s) defined for performing the column design is/are not selected based on the selected Inputs and/or "
1110 -                             "Design Preferences")
1111 -             logger.error("Cannot compute!")
1112 -             logger.info("Change the inputs provided and re-design.")
1113 -             self.design_status = False
1114 -             self.design_status_list.append(self.design_status)
1119 +         #else:
1120 +             # logger.warning("The section(s) defined for performing the column design is/are not selected based on the selected Inputs and/or "
1121 +                             # "Design Preferences")
1122 +             # logger.error("Cannot compute!")
1123 +             # logger.info("Change the inputs provided and re-design.")
1124 +             # self.design_status = False
1125 +             # self.design_status_list.append(self.design_status)
1126 +             # print(f"design_status_list{self.design_status_list}")
1115 1126
1116 1127
1117 1128     def results(self):
1118 1129         """ """
1119 -         # sorting results from the dataset
1120 -
1130 +         _ = [i for i in self.optimum_section_ur if i > 1.0]
1131 +         print( ' ',_)
1132 +         if len(_)==1:
1133 +             temp = _[0]
1134 +         elif len(_)==0:
1135 +             temp = None
1136 +         else:
1137 +             temp = sorted(_)[0]
1138 +         self.failed_design_dict = self.optimum_section_ur_results[temp] if temp is not None else None
1139 +         print('self.failed_design_dict ',self.failed_design_dict)
1140 +
1121 1141         # results based on UR
1122 1142         if self.optimization_parameter == 'Utilization Ratio':
1123 1143             filter_ur = filter(lambda x: x <= min(self.allowable_utilization_ratio, 1.0), self.optimum_section_ur)
1124 1144             self.optimum_section_ur = list(filter_ur)
1125 1145
1126 1146             self.optimum_section_ur.sort()
1127 -             # print(f"self.optimum_section_ur{self.optimum_section_ur}")
1128 -             #print(f"self.result_ur{self.result_ur}")
1147 +             print(f"self.optimum_section_ur{self.optimum_section_ur} \n self.optimum_section_ur_results{self.optimum_section_ur_results}")
1148 +
1149 +             # print(f"self.result_ur{self.result_ur}")
1129 1150
1130 1151             # selecting the section with most optimum UR
1131 1152             if len(self.optimum_section_ur) == 0: # no design was successful
1132 +             @@ -1134,138 +1155,166 @@ def results(self):
1134 1155                 logger.error("The solver did not find any adequate section from the defined list.")
1135 1156                 logger.info("Re-define the list of sections or check the Design Preferences option and re-design.")
1136 1157                 self.design_status = False
1137 -                 self.design_status_list.append(self.design_status)
1158 +                 if len(self.failed_design_dict)>0:
1159 +                     logger.info(

```

Figure 4.15: From Column.py

```

src/osdag/design_type/compression_member/Column.py
1147 + print(f"self.optimum_section_ur{self.optimum_section_ur} \n self.optimum_section_ur_results{self.optimum_section_ur_results}")
1148 +
1149 + # print(f"self.result_UR{self.result_UR}")
1129 1150
1130 1151 # selecting the section with most optimum UR
1131 1152 if len(self.optimum_section_ur) == 0: # no design was successful
1132 +
1133 + @@ -1134,138 +1155,166 @@ def results(self):
1134 1155     logger.error("The solver did not find any adequate section from the defined list.")
1135 1156     logger.info("Re-define the list of sections or check the Design Preferences option and re-design.")
1136 1157     self.design_status = False
1137 - self.design_status_list.append(self.design_status)
1138 +
1139 + if len(self.failed_design_dict)>0:
1140 +     logger.info(
1141 +         "The details for the best section provided is being shown"
1142 +     )
1143 +     self.result_UR = self.failed_design_dict['UR'] #temp
1144 +     self.common_result(
1145 +         self,
1146 +         list_result=self.failed_design_dict,
1147 +         result_type=None,
1148 +     )
1149 +     logger.warning(
1150 +         "Re-define the list of sections or check the Design Preferences option and re-design."
1151 +     )
1152 +     return
1153 +     #self.design_status_list.append(self.design_status)
1138 1173
1139 1174 else:
1140 - self.result_UR = self.optimum_section_ur[-1] # optimum section which passes the UR check
1141 + self.failed_design_dict = None
1142 + self.result_UR = self.optimum_section_ur[
1143 +     -1
1144 + ] # optimum section which passes the UR check
1145 1179 print(f"self.result_UR{self.result_UR}")
1146 1180 self.design_status = True
1147 + self.common_result(
1148 +     self,
1149 +     list_result=self.optimum_section_ur_results,
1150 +     result_type=self.result_UR,
1151 + )
1143 1186
1144 1187 else: # results based on cost
1145 1188     self.optimum_section_cost.sort()
1146 1189
1147 1190 # selecting the section with most optimum cost
1148 1191 self.result_cost = self.optimum_section_cost[0]
1149 -
1150 + self.design_status = True
1151 1193 # print results
1152 - if len(self.optimum_section_ur) == 0:
1153 -     logger.warning(
1154 -         "The sections selected by the solver from the defined list of sections did not satisfy the Utilization Ratio (UR) "
1155 -         "..."

```

Figure 4.16: From Column.py

```

1351 +
1352 + def common_result(self, list_result, result_type):
1353 +
1354 +     self.result_designation = list_result[result_type]['Designation']
1355 +     self.section_class = self.input_section_classification[self.result_designation][0]
1356 +
1357 +     if self.section_class == 'Slender':
1358 +         logger.warning("The trial section ({} is Slender. Computing the Effective Sectional Area as per Sec. 9.7.2, "
1359 +             "Fig. 2 (B & C) of The National Building Code of India (NBC), 2016.".format(self.result_designation))
1360 +         if self.effective_area_factor < 1.0:
1361 +             self.effective_area = round(self.effective_area * self.effective_area_factor, 2)
1362 +
1363 +         logger.warning("Reducing the effective sectional area as per the definition in the Design Preferences tab.")
1364 +         logger.info("The actual effective area is {} mm2 and the reduced effective area is {} mm2 [Reference: Cl. 7.3.2, IS 800:2007]".
1365 +             format(round(self.effective_area / self.effective_area_factor), 2), self.effective_area)

```

Figure 4.17: From Column.py

```

src/osdag/design_type/compression_member/Column.py
1367 +         if self.section_class != 'Slender':
1368 +             logger.info("The effective sectional area is taken as 100% of the cross-sectional area [Reference: Cl. 7.3.2, IS 800:2007].")
1369 +             logger.info(
1370 +                 "The section is {}. The {} section has {} flange({}) and {} web({}). [Reference: Cl 3.7, IS 800:2007].".format(
1371 +                     self.input_section_classification[self.result_designation][0] ,
1372 +                     self.result_designation,
1373 +                     self.input_section_classification[self.result_designation][1], round(self.input_section_classification[self.result_designation][3],2),
1374 +                     self.input_section_classification[self.result_designation][2], round(self.input_section_classification[self.result_designation][4],2)
1375 +                 ))
1411 1376
1377 +         self.result_section_class = list_result[result_type]['Section class']
1378 +         self.result_effective_area = list_result[result_type]['Effective area']
1379 +
1380 +         self.result_bc_zz = list_result[result_type]['Buckling_curve_zz']
1381 +         self.result_bc_yy = list_result[result_type]['Buckling_curve_yy']
1382 +
1383 +         self.result_IF_zz = list_result[result_type]['IF_zz']
1384 +         self.result_IF_yy = list_result[result_type]['IF_yy']
1385 +
1386 +         self.result_eff_len_zz = list_result[result_type]['Effective_length_zz']
1387 +         self.result_eff_len_yy = list_result[result_type]['Effective_length_yy']
1388 +
1389 +         self.result_eff_sr_zz = list_result[result_type]['Effective_SR_zz']
1390 +         self.result_eff_sr_yy = list_result[result_type]['Effective_SR_yy']
1391 +
1392 +         self.result_ebs_zz = list_result[result_type]['EBS_zz']
1393 +         self.result_ebs_yy = list_result[result_type]['EBS_yy']
1394 +
1395 +         self.result_nd_esr_zz = list_result[result_type]['ND_ESR_zz']
1396 +         self.result_nd_esr_yy = list_result[result_type]['ND_ESR_yy']
1397 +
1398 +         self.result_phi_zz = list_result[result_type]['phi_zz']
1399 +         self.result_phi_yy = list_result[result_type]['phi_yy']
1400 +
1401 +         self.result_srf_zz = list_result[result_type]['SRF_zz']
1402 +         self.result_srf_yy = list_result[result_type]['SRF_yy']
1403 +
1404 +         self.result_fcd_1_zz = list_result[result_type]['FCD_1_zz']
1405 +         self.result_fcd_1_yy = list_result[result_type]['FCD_1_yy']
1406 +         self.result_fcd_2 = list_result[result_type]['FCD_2']
1407 +         self.result_fcd_zz = list_result[result_type]['FCD_zz']
1408 +         self.result_fcd_yy = list_result[result_type]['FCD_yy']
1409 +         self.result_fcd = list_result[result_type]['FCD']
1410 +
1411 +         self.result_capacity = list_result[result_type]['Capacity']
1412 +         self.result_cost = list_result[result_type]['Cost']
1413 +
1414 +
1415 +         def save_design(self, popup_summary):
1416 +
1417 +             if (self.design_status and self.failed_design_dict is None) or (not self.design_status and len(self.failed_design_dict)>0):
1418 +                 if self.sec_profile=='Columns' or self.sec_profile=='Beams' or self.sec_profile == VALUES_SEC_PROFILE[0]:
1419 +                     self.report_column = {KEY_DISP_SEC_PROFILE: "ISection",

```

Figure 4.18: From Column.py


```

src/osdag/design_type/compression_member/Column.py
1415 +     def save_design(self, popup_summary):
1416 +
1417 +         if (self.design_status and self.failed_design_dict is None) or (not self.design_status and len(self.failed_design_dict)>0):
1418 +             if self.sec_profile=='Columns' or self.sec_profile=='Beams' or self.sec_profile == VALUES_SEC_PROFILE[0]:
1419 +                 self.report_column = {KEY_DISP_SEC_PROFILE: "ISection",
1420 +                                     KEY_DISP_SECSIZE: (self.section_property.designation, self.sec_profile),
1421 +                                     KEY_DISP_COLSEC_REPORT: self.section_property.designation,
1422 +                                     KEY_DISP_MATERIAL: self.section_property.material,
1423 +                                     # KEY_DISP_APPLIED_AXIAL_FORCE: self.section_property.,
1424 +                                     KEY_REPORT_MASS: self.section_property.mass,
1425 +                                     KEY_REPORT_AREA: round(self.section_property.area * 1e-2, 2),
1426 +                                     KEY_REPORT_DEPTH: self.section_property.depth,
1427 +                                     KEY_REPORT_WIDTH: self.section_property.flange_width,
1428 +                                     KEY_REPORT_WEB_THK: self.section_property.web_thickness,
1429 +                                     KEY_REPORT_FLANGE_THK: self.section_property.flange_thickness,
1430 +                                     KEY_DISP_FLANGE_S_REPORT: self.section_property.flange_slope,
1431 +                                     KEY_REPORT_R1: self.section_property.root_radius,
1432 +                                     KEY_REPORT_R2: self.section_property.toe_radius,
1433 +                                     KEY_REPORT_IZ: round(self.section_property.mom_inertia_z * 1e-4, 2),
1434 +                                     KEY_REPORT_IY: round(self.section_property.mom_inertia_y * 1e-4, 2),
1435 +                                     KEY_REPORT_RZ: round(self.section_property.rad_of_gy_z * 1e-1, 2),
1436 +                                     KEY_REPORT_RY: round(self.section_property.rad_of_gy_y * 1e-1, 2),
1437 +                                     KEY_REPORT_ZEZ: round(self.section_property.elast_sec_mod_z * 1e-3, 2),
1438 +                                     KEY_REPORT_ZEY: round(self.section_property.elast_sec_mod_y * 1e-3, 2),
1439 +                                     KEY_REPORT_ZPZ: round(self.section_property.plast_sec_mod_z * 1e-3, 2),
1440 +                                     KEY_REPORT_ZPY: round(self.section_property.plast_sec_mod_y * 1e-3, 2)}
1441 +
1442 +             else:
1443 +                 #Update for section profiles RHS and SHS, CHS by making suitable elif condition.
1444 +                 self.report_column = {KEY_DISP_COLSEC_REPORT: self.section_property.designation,
1445 +                                     KEY_DISP_MATERIAL: self.section_property.material,
1446 +                                     # KEY_DISP_APPLIED_AXIAL_FORCE: self.section_property.,
1447 +                                     KEY_REPORT_MASS: self.section_property.mass,
1448 +                                     KEY_REPORT_AREA: round(self.section_property.area * 1e-2, 2),
1449 +                                     KEY_REPORT_DEPTH: self.section_property.depth,
1450 +                                     KEY_REPORT_WIDTH: self.section_property.flange_width,
1451 +                                     KEY_REPORT_WEB_THK: self.section_property.web_thickness,
1452 +                                     KEY_REPORT_FLANGE_THK: self.section_property.flange_thickness,
1453 +                                     KEY_DISP_FLANGE_S_REPORT: self.section_property.flange_slope}
1454 +
1455 +             self.report_input = \
1456 +                 {#KEY_MAIN_MODULE: self.mainmodule,
1457 +                 KEY_MODULE: self.module, #"Axial load on column "
1458 +                 KEY_DISP_AXIAL: self.load.axial_force * 10 ** -3,
1459 +                 KEY_DISP_ACTUAL_LEN_ZZ: self.length_zz,
1460 +                 KEY_DISP_ACTUAL_LEN_YY: self.length_yy,
1461 +                 KEY_DISP_SEC_PROFILE: self.sec_profile,
1462 +                 KEY_DISP_SECSIZE: self.result_section_class,
1463 +                 KEY_DISP_END1: self.end_1_z,
1464 +                 KEY_DISP_END2: self.end_2_z,
1465 +                 KEY_DISP_END1_Y: self.end_1_y,
1466 +                 KEY_DISP_END2_Y: self.end_2_y,
1467 +                 "Column Section - Mechanical Properties": "TITLE"}

```

Figure 4.19: From Column.py

```

src/osdag/design_type/compression_member/Column.py  +491 -349
1453 +
1454 +     self.report_input = \
1455 +         {#KEY_MAIN_MODULE: self.mainmodule,
1456 +         KEY_MODULE: self.module, # "Axial load on column "
1457 +         KEY_DISP_AXIAL: self.load.axial_force * 10 ** -3,
1458 +         KEY_DISP_ACTUAL_LEN_ZZ: self.length_zz,
1459 +         KEY_DISP_ACTUAL_LEN_YY: self.length_yy,
1460 +         KEY_DISP_SEC_PROFILE: self.sec_profile,
1461 +         KEY_DISP_SECSIZE: self.result.section_class,
1462 +         KEY_DISP_END1: self.end_1_z,
1463 +         KEY_DISP_END2: self.end_2_z,
1464 +         KEY_DISP_END1_Y: self.end_1_y,
1465 +         KEY_DISP_END2_Y: self.end_2_y,
1466 +         "Column Section - Mechanical Properties": "TITLE",
1467 +         KEY_MATERIAL: self.material,
1468 +         KEY_DISP_ULTIMATE_STRENGTH_REPORT: self.material_property.fu,
1469 +         KEY_DISP_YIELD_STRENGTH_REPORT: self.material_property.fy,
1470 +         KEY_DISP_EFFECTIVE_AREA_PARA: self.effective_area_factor, #To Check
1471 +         KEY_DISP_SECSIZE: str(self.sec_list),
1472 +         "Selected Section Details": self.report_column,
1473 +         }
1474 +
1475 +
1476 +     self.report_check = []
1477 +     t1 = ('Selected', 'Selected Member Data', '|p{5cm}|p{2cm}|p{2cm}|p{2cm}|p{4cm}|')
1478 +     self.report_check.append(t1)
1479 +
1480 +     self.h = (self.section_property.depth - 2 * (self.section_property.flange_thickness + self.section_property.root_radius))
1481 +     self.h_bf_ratio = self.h / self.section_property.flange_width
1482 +
1483 +
1484 +     # 2.2 CHECK: Buckling Class - Compatibility Check
1485 +     t1 = ('SubSection', 'Buckling Class - Compatibility Check', '|p{4cm}|p{3.5cm}|p{6.5cm}|p{2cm}|')
1486 +     self.report_check.append(t1)
1487 +
1488 +     # YY axis row
1489 +     t1 = (
1490 +         "h/bf and tf for YY Axis",
1491 +         comp_column_class_section_check_required(self.h, self.section_property.flange_width, self.section_property.flange_thickness, "YY"),
1492 +         comp_column_class_section_check_provided(self.h, self.section_property.flange_width, self.section_property.flange_thickness, round(self.h_bf_ratio, 2), "YY"),
1493 +         'Compatible'
1494 +     )
1495 +     self.report_check.append(t1)
1496 +
1497 +     # ZZ axis row
1498 +     t1 = (
1499 +         "h/bf and tf for ZZ Axis",
1500 +         comp_column_class_section_check_required(self.h, self.section_property.flange_width, self.section_property.flange_thickness, "ZZ"),
1501 +         comp_column_class_section_check_provided(self.h, self.section_property.flange_width, self.section_property.flange_thickness, round(self.h_bf_ratio, 2), "ZZ"),
1502 +         'Compatible'
1503 +     )
1504 +     self.report_check.append(t1)
1505 +
1506 +

```

Figure 4.20: From Column.py

```

src/osdag/design_type/compression_member/Column.py  +
1506 +
1507 +     t1 = ('SubSection', 'Section Classification', '|p{3cm}|p{3.5cm}|p{8.5cm}|p{1cm}|')
1508 +     self.report_check.append(t1)
1509 +     t1 = ('Web Class', 'Axial Compression',
1510 +         cl_3_7_2_section_classification_web(round(self.h, 2), round(self.section_property.web_thickness, 2),
1511 +         round(self.input_section_classification[self.result_designation][4], 2),
1512 +         self.epsilon, self.section_property.type,
1513 +         self.input_section_classification[self.result_designation][2]),
1514 +         ' ')
1515 +     self.report_check.append(t1)
1516 +     t1 = ('Flange Class', self.section_property.type,
1517 +         cl_3_7_2_section_classification_flange(round(self.section_property.flange_width/2, 2),
1518 +         round(self.section_property.flange_thickness, 2),
1519 +         round(self.input_section_classification[self.result_designation][3], 2),
1520 +         self.epsilon,
1521 +         self.input_section_classification[self.result_designation][1]),
1522 +         ' ')
1523 +     self.report_check.append(t1)
1524 +     t1 = ('Section Class', ' ',
1525 +         cl_3_7_2_section_classification(
1526 +         self.input_section_classification[self.result_designation][0]),
1527 +         ' ')
1528 +     self.report_check.append(t1)
1529 +
1530 +     t1 = ('NewTable', 'Imperfection Factor', '|p{3cm}|p{5 cm}|p{5cm}|p{3 cm}|')
1531 +     self.report_check.append(t1)
1532 +
1533 +     t1 = (
1534 +         'YY',
1535 +         self.list_yy[3].upper(),
1536 +         self.list_yy[4], ' '
1537 +     )
1538 +     self.report_check.append(t1)
1539 +
1540 +     t1 = (
1541 +         'ZZ',
1542 +         self.list_zz[3].upper(),
1543 +         self.list_zz[4], ' '
1544 +     )
1545 +     self.report_check.append(t1)
1546 +
1547 +     K_yy = self.result_eff_len_yy / self.length_yy
1548 +     K_zz = self.result_eff_len_zz / self.length_zz
1549 +     t1 = ('SubSection', 'Slenderness Ratio', '|p{4cm}|p{2 cm}|p{7cm}|p{3 cm}|')
1550 +     self.report_check.append(t1)
1551 +     t1 = ("Effective Slenderness Ratio (For YY Axis)", ' ',
1552 +         cl_7_1_2_effective_slenderness_ratio(K_yy, self.length_yy, self.section_property.rad_of_gy_y, round(self.result_eff_sr_yy, 2)),
1553 +         ' ')
1554 +     self.report_check.append(t1)
1555 +     t1 = ("Effective Slenderness Ratio (For ZZ Axis)", ' ',
1556 +         cl_7_1_2_effective_slenderness_ratio(K_zz, self.length_zz, self.section_property.rad_of_gy_z, round(self.result_eff_sr_zz, 2)),
1557 +         ' ')

```

Figure 4.21: From Column.py

```

src/osdag/design_type/compression_member/Column.py 491-349
1537 +         t1 = (
1538 +             'ZZ',
1539 +             self.list_zz[3].upper(),
1540 +             self.list_zz[4], ''
1541 +         )
1542 +         self.report_check.append(t1)
1543 +
1544 +         K_yy = self.result_eff_len_yy / self.length_yy
1545 +         K_zz = self.result_eff_len_zz / self.length_zz
1546 +         t1 = ('SubSection', 'Slenderness Ratio', '|p{4cm}|p{2 cm}|p{7cm}|p{3 cm}|')
1547 +         self.report_check.append(t1)
1548 +         t1 = ('Effective Slenderness Ratio (For YY Axis)', '',
1549 +             c1_7_1_2_effective_slenderness_ratio(K_yy, self.length_yy, self.section_property.rad_of_gy_y, round(self.result_eff_sr_yy, 2)),
1550 +             ''
1551 +         )
1552 +         self.report_check.append(t1)
1553 +         t1 = ('Effective Slenderness Ratio (For ZZ Axis)', '',
1554 +             c1_7_1_2_effective_slenderness_ratio(K_zz, self.length_zz, self.section_property.rad_of_gy_z, round(self.result_eff_sr_zz, 2)),
1555 +             ''
1556 +         )
1557 +         self.report_check.append(t1)
1558 +
1559 +         t1 = ('SubSection', 'Checks', '|p{4cm}|p{2 cm}|p{7cm}|p{3 cm}|')
1560 +         self.report_check.append(t1)
1561 +
1562 +         t1 = (r'$\phi_{yy}$', '',
1563 +             c1_8_7_1_5_phi(self.result_IF_yy, round(self.non_dim_eff_sr_yy, 2), round(self.result_phi_yy, 2)),
1564 +             ''
1565 +         )
1566 +         self.report_check.append(t1)
1567 +
1568 +         t1 = (r'$\phi_{zz}$', '',
1569 +             c1_8_7_1_5_phi(self.result_IF_zz, round(self.non_dim_eff_sr_zz, 2), round(self.result_phi_zz, 2)),
1570 +             ''
1571 +         )
1572 +         self.report_check.append(t1)
1573 +         t1 = (r'$F_{cd,yy}$ \, \left( \frac{N}{\text{mm}^2} \right)$', '',
1574 +             c1_8_7_1_5_Buckling(self.material_property.fy, self.gamma_m0, round(self.non_dim_eff_sr_yy, 2), round(self.result_phi_yy, 2), round(self.result_fcd_2, 2),
1575 +             round(self.result_fcd_yy, 2)),
1576 +             ''
1577 +         )
1578 +         self.report_check.append(t1)
1579 +         t1 = (r'$F_{cd,zz}$ \, \left( \frac{N}{\text{mm}^2} \right)$', '',
1580 +             c1_8_7_1_5_Buckling(self.material_property.fy, self.gamma_m0, round(self.non_dim_eff_sr_zz, 2), round(self.result_phi_zz, 2), round(self.result_fcd_2, 2),
1581 +             round(self.result_fcd_zz, 2)),
1582 +             ''
1583 +         )
1584 +         self.report_check.append(t1)
1585 +         t1 = (r'Design Compressive Strength \(( P_d \)) (For the most critical value of \(( F_{cd} \))\)', self.load.axial_force * 10 ** -3,
1586 +             c1_7_1_2_design_compressive_strength(round(self.result_capacity / 1000, 2), self.section_property.area, round(self.result_fcd, 2), self.load.axial_force * 10
1587 +             ** -3),
1588 +             get_pass_fail(self.load.axial_force * 10 ** -3, round(self.result_capacity, 2), relation='leq'))
1589 +         self.report_check.append(t1)

```

Figure 4.22: From Column.py

```

1588 +         else:
1589 +             self.report_input = \
1590 +                 {#KEY_MAIN_MODULE: self.mainmodule,
1591 +                 KEY_MODULE: self.module, #Axial load on column "
1592 +                 KEY_DISP_AXIAL: self.load.axial_force * 10 ** -3,
1593 +                 KEY_DISP_ACTUAL_LEN_ZZ: self.length_zz,
1594 +                 KEY_DISP_ACTUAL_LEN_YY: self.length_yy,
1595 +                 KEY_DISP_SEC_PROFILE: self.sec_profile,
1596 +                 KEY_DISP_SECSIZE: str(self.sec_list),
1597 +                 #KEY_DISP_SECSIZE: self.result_section_class,
1598 +                 KEY_DISP_END1: self.end_1_z,
1599 +                 KEY_DISP_END2: self.end_2_z,
1600 +                 KEY_DISP_END1_Y: self.end_1_y,
1601 +                 KEY_DISP_END2_Y: self.end_2_y,
1602 +                 "Column Section - Mechanical Properties": "TITLE",
1603 +                 KEY_MATERIAL: self.material,
1604 +                 KEY_DISP_ULTIMATE_STRENGTH_REPORT: self.material_property.fu,
1605 +                 KEY_DISP_YIELD_STRENGTH_REPORT: self.material_property.fy,
1606 +                 KEY_DISP_EFFECTIVE_AREA_PARA: self.effective_area_factor, #To Check
1607 +                 # "Failed Section Details": self.report_column,
1608 +                 }
1609 +             self.report_check = []
1610 +
1611 +             t1 = ('Selected', 'All Members Failed', '|p{5cm}|p{2cm}|p{2cm}|p{2cm}|p{4cm}|')
1612 +             self.report_check.append(t1)
1613 +
1614 +
1615 +
1616 +         Disp_2d_image = []
1617 +         @@ -1481,4 +1622,5 @@ def save_design(self, popup_summary):
1618 +             rel_path = rel_path.replace("\\", "/")
1619 +             fname_no_ext = popup_summary['filename']
1620 +             CreateLatex.save_latex(CreateLatex(), self.report_input, self.report_check, popup_summary, fname_no_ext,
1621 +             rel_path, Disp_2d_image, Disp_3D_image, module=self.module)
1622 +             rel_path, Disp_2d_image, Disp_3D_image, module=self.module)
1623 +
1624 +

```

Figure 4.23: From Column.py

```

+
+   Disp_2d_image = []
+   Disp_3D_image = "/ResourceFiles/images/3d.png"
+
    print(sys.path[0])
    rel_path = str(sys.path[0])
    rel_path = os.path.abspath(".") # TEMP
    rel_path = rel_path.replace("\\", "/")
    fname_no_ext = popup_summary['filename']
    CreateLatex.save_latex(CreateLatex(), self.report_input, self.report_check, popup_summary, fname_no_ext,
-                               rel_path, module=self.module)
-
-
-
+                               rel_path, Disp_2d_image, Disp_3D_image, module=self.module)

```

Figure 4.24: From Column.py

4.3.4 compression.py

In the `compression.py`, `section_classification` and `save_design` functions were modified, a failure dictionary was added as well to fall back to for report generation when `design_status` fails. The last 4 images were integrated as well similar to `compression.py`.

```

src/osdag/design_type/compression_member/compression.py
11 11 from ...utils.common import is800_2007
12 12 from ...utils.common.component import *
13 13
14 +import logging
15 +from ..connection.moment_connection import MomentConnection
16 +from ...utils.common.material import *
17 +from ...Report_functions import *
18 +from ...design_report.reportGenerator_latex import CreateLatex
19 +from pylatex.utils import NoEscape
14 20
15 21
16 22 class Compression(Member):
229 229 #####
230 230
231 231 def module_name(self):
232 - return KEY_DISP_COMPRESSION_STRUT
233 + return KEY_DISP_COMPRESSION_Strut
234 234
235 235 def set_osdaglogger(key):
236 236
237 237
238 238
239 239
240 240
241 241
242 242
243 243
244 244
245 245
246 246
247 247
248 248
249 249
250 250
251 251
252 252
253 253
254 254
255 255
256 256
257 257
258 258
259 259
260 260
261 261
262 262
263 263
264 264
265 265
266 266
267 267
268 268
269 269
270 270
271 271
272 272
273 273
274 274
275 275
276 276
277 277
278 278
279 279
280 280
281 281
282 282
283 283
284 284
285 285
286 286
287 287
288 288
289 289
290 290
291 291
292 292
293 293
294 294
295 295
296 296
297 297
298 298
299 299
300 300
301 301
302 302
303 303
304 304
305 305
306 306
307 307
308 308
309 309
310 310
311 311
312 312
313 313
314 314
315 -
316 + self.module = KEY_DISP_COMPRESSION_Strut
317 options_list = []
318
319
320 # @author: Amir, Umair
321
322
323
324 t1 = (KEY_MODULE, KEY_DISP_COMPRESSION_Strut, TYPE_MODULE, None, True, 'No Validator')
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840 846 return all_errors
841 847 print(f"func_for_validation done")
842 848
849 +
850 + def get_3d_components(self):
851 +
852 + components = []
853 +
854 + t1 = ('Model', self.call_3DModel)
855 + components.append(t1)
856 +
857 + return components
858 +
859
860
861
862
863 859 def fn_conn_type(self):
864 860
865 861 "Function to populate section size based on the type of section "
866 862
867 863
868 864
869 865
870 866
871 867
872 868
873 869
874 870
875 871
876 872
877 873
878 874
879 875
880 876
881 877
882 878
883 879
884 880
885 881
886 882
887 883
888 884
889 885
890 886
891 887
892 888
893 889
894 890
895 891
896 892
897 893
898 894
899 895
900 896
901 897
902 898
903 899
904 900
905 901
906 902
907 903
908 904
909 905
910 906
911 907
912 908
913 909
914 910
915 911
916 912
917 913
918 914
919 915
920 916
921 917
922 918
923 919
924 920
925 921
926 922
927 923
928 924
929 925
930 926
931 927
932 928
933 929
934 930
935 931
936 932
937 933
938 934
939 935
940 936
941 937
942 938
943 939
944 940
945 941
946 942
947 943
948 944
949 945
950 946
951 947
952 948
953 949
954 950
955 951
956 952
957 953
958 954
959 955
960 956
961 957
962 958
963 959
964 960
965 961
966 962
967 963
968 964
969 965
970 966
971 967
972 968
973 969
974 970
975 971
976 972
977 973
978 974
979 975
980 976
981 977
982 978
983 979
984 980
985 981
986 982
987 983
988 984
989 985
990 986
991 987
992 988
993 989
994 990
995 991
996 992
997 993
998 994
999 995
1000 996
1001 997
1002 998
1003 999
1004 1000
1005 1001
1006 1002
1007 1003
1008 1004
1009 1005
1010 1006
1011 1007
1012 1008
1013 1009
1014 1010
1015 1011
1016 1012
1017 1013
1018 1014
1019 1015
1020 1016
1021 1017
1022 1018
1023 1019
1024 1020
1025 1021
1026 1022
1027 1023
1028 1024
1029 1025
1030 1026
1031 1027
1032 1028
1033 1029
1034 1030
1035 1031
1036 1032
1037 1033
1038 1034
1039 1035
1040 1036
1041 1037
1042 1038
1043 1039
1044 1040
1045 1041
1046 1042
1047 1043
1048 1044
1049 1045
1050 1046
1051 1047
1052 1048
1053 1049
1054 1050
1055 1051
1056 1052
1057 1053
1058 1054
1059 1055
1060 1056
1061 1057
1062 1058
1063 1059
1064 1060
1065 1061
1066 1062
1067 1063
1068 1064
1069 1065
1070 1066
1071 1067
1072 1068
1073 1069
1074 1070
1075 1071
1076 1072
1077 1073
1078 1074
1079 1075
1080 1076
1081 1077
1082 1078
1083 1079
1084 1080
1085 1081
1086 1082
1087 1083
1088 1084
1089 1085
1090 1086
1091 1087
1092 1088
1093 1089
1094 1090
1095 1091
1096 1092
1097 1093
1098 1094
1099 1095
1100 1096
1101 1097
1102 1098
1103 1099
1104 1100
1105 1101
1106 1102
1107 1103
1108 1104
1109 1105
1110 1106
1111 1107
1112 1108
1113 1109
1114 1110
1115 1111
1116 1112
1117 1113
1118 1114
1119 1115
1120 1116
1121 1117
1122 1118
1123 1119
1124 1120
1125 1121
1126 1122
1127 1123
1128 1124
1129 1125
1130 1126
1131 1127
1132 1128
1133 1129
1134 1130
1135 1131
1136 1132
1137 1133
1138 1134
1139 1135
1140 1136
1141 1137
1142 1138
1143 1139
1144 1140
1145 1141
1146 1142
1147 1143
1148 1144
1149 1145
1150 1146
1151 1147
1152 1148
1153 1149
1154 1150
1155 1151
1156 1152
1157 1153
1158 1154
1159 1155
1160 1156
1161 1157
1162 1158
1163 1159
1164 1160
1165 1161
1166 1162
1167 1163
1168 1164
1169 1165
1170 1166
1171 1167
1172 1168
1173 1169
1174 1170
1175 1171
1176 1172
1177 1173
1178 1174
1179 1175
1180 1176
1181 1177
1182 1178
1183 1179
1184 1180
1185 1181
1186 1182
1187 1183
1188 1184
1189 1185
1190 1186
1191 1187
1192 1188
1193 1189
1194 1190
1195 1191
1196 1192
1197 1193
1198 1194
1199 1195
1200 1196
1201 1197
1202 1198
1203 1199
1204 1200
1205 1201
1206 1202
1207 1203
1208 1204
1209 1205
1210 1206
1211 1207
1212 1208
1213 1209
1214 1210
1215 1211
1216 1212
1217 1213
1218 1214
1219 1215
1220 1216
1221 1217
1222 1218
1223 1219
1224 1220
1225 1221
1226 1222
1227 1223
1228 1224
1229 1225
1230 1226
1231 1227
1232 1228
1233 1229
1234 1230
1235 1231
1236 1232
1237 1233
1238 1234
1239 1235
1240 1236
1241 1237
1242 1238
1243 1239
1244 1240
1245 1241
1246 1242
1247 1243
1248 1244
1249 1245
1250 1246
1251 1247
1252 1248
1253 1249
1254 1250
1255 1251
1256 1252
1257 1253
1258 1254
1259 1255
1260 1256
1261 1257
1262 1258
1263 1259
1264 1260
1265 1261
1266 1262
1267 1263
1268 1264
1269 1265
1270 1266
1271 1267
1272 1268
1273 1269
1274 1270
1275 1271
1276 1272
1277 1273
1278 1274
1279 1275
1280 1276
1281 1277
1282 1278
1283 1279
1284 1280
1285 1281
1286 1282
1287 1283
1288 1284
1289 1285
1290 1286
1291 1287
1292 1288
1293 1289
1294 1290
1295 1291
1296 1292
1297 1293
1298 1294
1299 1295
1300 1296
1301 1297
1302 1298
1303 1299
1304 1300
1305 1301
1306 1302
1307 1303
1308 1304
1309 1305
1310 1306
1311 1307
1312 1308
1313 1309
1314 1310
1315 1311
1316 1312
1317 1313
1318 1314
1319 1315
1320 1316
1321 1317
1322 1318
1323 1319
1324 1320
1325 1321
1326 1322
1327 1323
1328 1324
1329 1325
1330 1326
1331 1327
1332 1328
1333 1329
1334 1330
1335 1331
1336 1332
1337 1333
1338 1334
1339 1335
1340 1336
1341 1337
1342 1338
1343 1339
1344 1340
1345 1341
1346 1342
1347 1343
1348 1344
1349 1345
1350 1346
1351 1347
1352 1348
1353 1349
1354 1350
1355 1351
1356 1352
1357 1353
1358 1354
1359 1355
1360 1356
1361 1357
1362 1358
1363 1359
1364 1360
1365 1361
1366 1362
1367 1363
1368 1364
1369 1365
1370 1366
1371 1367
1372 1368
1373 1369
1374 1370
1375 1371
1376 1372
1377 1373
1378 1374
1379 1375
1380 1376
1381 1377
1382 1378
1383 1379
1384 1380
1385 1381
1386 1382
1387 1383
1388 1384
1389 1385
1390 1386
1391 1387
1392 1388
1393 1389
1394 1390
1395 1391
1396 1392
1397 1393
1398 1394
1399 1395
1400 1396
1401 1397
1402 1398
1403 1399
1404 1400
1405 1401
1406 1402
1407 1403
1408 1404
1409 1405
1410 1406
1411 1407
1412 1408
1413 1409
1414 1410
1415 1411
1416 1412
1417 1413
1418 1414
1419 1415
1420 1416
1421 1417
1422 1418
1423 1419
1424 1420
1425 1421
1426 1422
1427 1423
1428 1424
1429 1425
1430 1426
1431 1427
1432 1428
1433 1429
1434 1430
1435 1431
1436 1432
1437 1433
1438 1434
1439 1435
1440 1436
1441 1437
1442 1438
1443 1439
1444 1440
1445 1441
1446 1442
1447 1443
1448 1444
1449 1445
1450 1446
1451 1447
1452 1448
1453 1449
1454 1450
1455 1451
1456 1452
1457 1453
1458 1454
1459 1455
1460 1456
1461 1457
1462 1458
1463 1459
1464 1460
1465 1461
1466 1462
1467 1463
1468 1464
1469 1465
1470 1466
1471 1467
1472 1468
1473 1469
1474 1470
1475 1471
1476 1472
1477 1473
1478 1474
1479 1475
1480 1476
1481 1477
1482 1478
1483 1479
1484 1480
1485 1481
1486 1482
1487 1483
1488 1484
1489 1485
1490 1486
1491 1487
1492 1488
1493 1489
1494 1490
1495 1491
1496 1492
1497 1493
1498 1494
1499 1495
1500 1496
1501 1497
1502 1498
1503 1499
1504 1500
1505 1501
1506 1502
1507 1503
1508 1504
1509 1505
1510 1506
1511 1507
1512 1508
1513 1509
1514 1510
1515 1511
1516 1512
1517 1513
1518 1514
1519 1515
1520 1516
1521 1517
1522 1518
1523 1519
1524 1520
1525 1521
1526 1522
1527 1523
1528 1524
1529 1525
1530 1526
1531 1527
1532 1528
1533 1529
1534 1530
1535 1531
1536 1532
1537 1533
1538 1534
1539 1535
1540 1536
1541 1537
1542 1538
1543 1539
1544 1540
1545 1541
1546 1542
1547 1543
1548 1544
1549 1545
1550 1546
1551 1547
1552 1548
1553 1549
1554 1550
1555 1551
1556 1552
1557 1553
1558 1554
1559 1555
1560 1556
1561 1557
1562 1558
1563 1559
1564 1560
1565 1561
1566 1562
1567 1563
1568 1564
1569 1565
1570 1566
1571 1567
1572 1568
1573 1569
1574 1570
1575 1571
1576 1572
1577 1573
1578 1574
1579 1575
1580 1576
1581 1577
1582 1578
1583 1579
1584 1580
1585 1581
1586 1582
1587 1583
1588 1584
1589 1585
1590 1586
1591 1587
1592 1588
1593 1589
1594 1590
1595 1591
1596 1592
1597 1593
1598 1594
1599 1595
1600 1596
1601 1597
1602 1598
1603 1599
1604 1600
1605 1601
1606 1602
1607 1603
1608 1604
1609 1605
1610 1606
1611 1607
1612 1608
1613 1609
1614 1610
1615 1611
1616 1612
1617 1613
1618 1614
1619 1615
1620 1616
1621 1617
1622 1618
1623 1619
1624 1620
1625 1621
1626 1622
1627 1623
1628 1624
1629 1625
1630 1626
1631 1627
1632 1628
1633 1629
1634 1630
1635 1631
1636 1632
1637 1633
1638 1634
1639 1635
1640 1636
1641 1637
1642 1638
1643 1639
1644 1640
1645 1641
1646 1642
1647 1643
1648 1644
1649 1645
1650 1646
1651 1647
1652 1648
1653 1649
1654 1650
1655 1651
1656 1652
1657 1653
1658 1654
1659 1655
1660 1656
1661 1657
1662 1658
1663 1659
1664 1660
1665 1661
1666 1662
1667 1663
1668 1664
1669 1665
1670 1666
1671 1667
1672 1668
1673 1669
1674 1670
1675 1671
1676 1672
1677 1673
1678 1674
1679 1675
1680 1676
1681 1677
1682 1678
1683 1679
1684 1680
1685 1681
1686 1682
1687 1683
1688 1684
1689 1685
1690 1686
1691 1687
1692 1688
1693 1689
1694 1690
1695 1691
1696 1692
1697 1693
1698 1694
1699 1695
1700 1696
1701 1697
1702 1698
1703 1699
1704 1700
1705 1701
1706 1702
1707 1703
1708 1704
1709 1705
1710 1706
1711 1707
1712 1708
1713 1709
1714 1710
1715 1711
1716 1712
1717 1713
1718 1714
1719 1715
1720 1716
1721 1717
1722 1718
1723 1719
1724 1720
1725 1721
1726 1722
1727 1723
1728 1724
1729 1725
1730 1726
1731 1727
1732 1728
1733 1729
1734 1730
1735 1731
1736 1732
1737 1733
1738 1734
1739 1735
1740 1736
1741 1737
1742 1738
1743 1739
1744 1740
1745 1741
1746 1742
1747 1743
1748 1744
1749 1745
1750 1746
1751 1747
1752 1748
1753 1749
1754 1750
1755 1751
1756 1752
1757 1753
1758 1754
1759 1755
1760 1756
1761 1757
1762 1758
1763 1759
1764 1760
1765 1761
1766 1762
1767 1763
1768 1764
1769 1765
1770 1766
1771 1767
1772 1768
1773 1769
1774 1770
1775 1771
1776 1772
1777 1773
1778 1774
1779 1775
1780 1776
1781 1777
1782 1778
1783 1779
1784 1780
1785 1781
1786 1782
1787 1783
1788 1784
1789 1785
1790 1786
1791 1787
1792 1788
1793 1789
1794 1790
1795 1791
1796 1792
1797 1793
1798 1794
1799 1795
1800 1796
1801 1797
1802 1798
1803 1799
1804 1800
1805 1801
1806 1802
1807 1803
1808 1804
1809 1805
1810 180
```

```

src/osdag/design_type/compression_member/compression.py
@@ -840,9 +839,21 @@ def func_for_validation(self, design_dictionary):
840 839         flag = True
841 840
842 841         #print(f'flag = {flag}')
843 -         if flag and flag1 and flag2:
842 +         if flag:
843 +             print(f"\n design_dictionary{design_dictionary}")
844 844             self.set_input_values(self, design_dictionary)
845 -         # print(design_dictionary)
845 +         if self.design_status == False and len(self.failed_design_dict) > 0:
846 +             logger.error(
847 +                 "Design Failed, Check Design Report"
848 +             )
849 +             return # ['Design Failed, Check Design Report'] @TODO
850 +             elif self.design_status:
851 +                 pass
852 +             else:
853 +                 logger.error(
854 +                     "Design Failed. Slender Sections Selected"
855 +                 )
856 +                 return # ['Design Failed. Slender Sections Selected']
846 857         else:
847 858             return all_errors
848 859         print(f"func_for_validation done")
@@ -872,7 +883,6 @@ def set_input_values(self, design_dictionary):
872 883         super(Compression,self).set_input_values(self, design_dictionary)
873 884         #self.sizelist == self.sec_list
874 885         # section properties
875 -         self.mainmodule = 'Struts in Trusses'
876 886         self.module = design_dictionary[KEY_MODULE]
877 887         self.mainmodule = 'Struts in Trusses'
878 888         self.sizelist = design_dictionary[KEY_SECSIZE]
@@ -1005,6 +1015,7 @@ def set_input_values(self, design_dictionary):
1005 1015         print("K = {}. \n The input values are set. Performing preliminary member check(s)".format(self.K))
1006 1016         # self.i = 0
1007 1017         # checking input values
1018 +         self.failed_design_dict = {}
1008 1019         flag = self.section_classification(self)
1009 1020         if len(self.input_section_list) == 0:
1010 1021             flag == False
@@ -1225,85 +1236,78 @@ def set_input_values(self, design_dictionary):
1225 1236         #         logger.error(": Design is unsafe. \n ")
1226 1237         #         logger.info(" :=====End Of design=====")
1227 1238
1239 +
1228 1240         def section_classification(self):
1229 -         """ Classify the sections based on Table 2 of IS 800:2007 """
1230 -         # print(f"Inside section_classification")
1231 -         first_section_logged = False

```

Figure 4.26: From compression.py

```

src/osdag/design_type/compression_member/compression.py
1018 + self.failed_design_dict = {}
1008 1019 flag = self.section_classification(self)
1009 1020 if len(self.input_section_list) == 0:
1010 1021     flag == False
@@ -1225,85 +1236,78 @@ def set_input_values(self, design_dictionary):
1225 1236 #         logger.error(": Design is unsafe. \n ")
1226 1237 #         logger.info(" :=====End Of design=====")
1227 1238
1239 +
1228 1240     def section_classification(self):
1229 -         """ Classify the sections based on Table 2 of IS 800:2007 """
1230 -         # print(f"Inside section_classification")
1231 -         first_section_logged = False
1241 +         """Classify the sections based on Table 2 of IS 800:2007"""
1242 +         print(f"Inside section_classification")
1232 1243         local_flag = True
1244 +         self.input_modified = []
1233 1245         self.input_section_list = []
1234 1246         self.input_section_classification = {}
1247 +         lambda_check = False
1235 1248
1236 -
1237 1249         for section in self.sec_list:
1238 1250             trial_section = section.strip("")
1239 1251             # print(f"trial_section {trial_section}")
1240 1252
1241 1253             # section_classification_subchecks(trial_section, self.material)
1242 1254
1243 -             # Fetch the section properties
1244 -             self.section_property = self.section_classification_subchecks(self, trial_section)
1245 -             # print(f"Type of section(type(section))")
1246 +             # fetching the section properties
1247 +             self.section_property = self.section_classification_subchecks(self, trial_section)
1248 +             print(f"Type of section(type(section))")
1249 1258
1250 +             # section classification
1247 1260             if (self.sec_profile in VALUES_SEC_PROFILE.Compression_Strut[:3]): # Angles or Back to Back or 'Star Angle'
1261 +
1248 1262                 # updating the material property based on thickness of the thickest element
1249 1263                 self.material_property.connect_to_database_to_get_fy_fu(self.material, self.section_property.thickness)
1250 -
1251 1264                 if self.section_property.type == 'Rolled':
1252 1265                     if self.sec_profile == VALUES_SEC_PROFILE.Compression_Strut[0] or self.sec_profile == VALUES_SEC_PROFILE.Compression_Strut[2]:
1253 -                         list_Table2_vi = IS800_2007.Table2_vi(self.section_property.min_leg, self.section_property.max_leg, self.section_property.thickness,
1266 +                         list_Table2_vi = IS800_2007.Table2_vi(self.section_property.min_leg, self.section_property.max_leg, self.section_property.thickness,
1254 1267                             self.material_property.fy, "Axial Compression")
1255 1268                     elif self.sec_profile == VALUES_SEC_PROFILE.Compression_Strut[1]:
1256 1269                         list_Table2_vii = IS800_2007.Table2_vii(self.section_property.min_leg,
1257 -                             self.section_property.max_leg,
1258 -                             self.section_property.thickness,
1259 -                             self.material_property.fy, "Axial Compression")
1270 +                         self.section_property.max_leg.

```

Figure 4.27: From compression.py

```

src/osdag/design_type/compression_member/compression.py
1387 - if self.section_property.section_class in self.allowed_sections and local_flag == True:
1391 + if self.section_property.section_class in self.allowed_sections and local_flag == True:
1388 1392     self.input_section_list.append(trial_section)
1389 1393     self.input_section_classification.update({trial_section: self.section_property.section_class})
1390 1394     # if self.sec_profile != Profile_name_1:
1391 1395     self.sec_prop_initial_dict.update({trial_section : (self.section_property.section_class, min_radius_gyration, slenderness, effective_area)}) #,
self.width_thickness_ratio,self.depth_thickness_ratio,self.width_depth_thickness_ratio
1392 1396     # print(f" sectopn class done {self.sec_list}")
1393 1397     return local_flag
1394 1398     # print(f"self.section_property.section_class{self.section_property.section_class}")
1399 +
1395 1400
1396 1401     # =====Calculations start here===== #
1397 1402     def optimization_tab_check(self):
@@ -1566,7 +1571,22 @@ def common_checks_1(self, section, step = 1, list_result = [], list_1 = []):
1566 1571
1567 1572
1568 1573     def common_result(self, list_result,result_type):
1569 - self.result_designation = list_result[result_type]['Designation']
1574 +
1575 + self.result_designation = list_result[result_type]['Designation'] # TODO debug
1576 + limit = IS800_2007.cl_3_8_max_slenderness_ratio(1)
1577 + if self.sec_prop_initial_dict[self.result_designation][2] > limit:
1578 +     logger.warning("Length provided is beyond the limit allowed. [Reference: Cl 3.8, IS 800:2007]")
1579 +     logger.error("Cannot compute. Given Length does not pass for this section.")
1580 +     # self.sec_list.remove(self.section_property.designation )
1581 + else:
1582 +     logger.info("Length provided is within the limit allowed. [Reference: Cl 3.8, IS 800:2007]" )
1583 +     logger.info(
1584 +         "The section is {}. The b/t of the section ({} ) is {} and d/t is {} and (b+d)/t is {}. [Reference: Cl 3.7, IS 800:2007]".format(
1585 +             self.input_section_classification[self.result_designation], self.result_designation,
1586 +             round(self.width_thickness_ratio, 2), round_up(self.depth_thickness_ratio,
1587 +                 round(self.width_depth_thickness_ratio, 2)))
1588 +
1589 +
1590 + self.result_section_class = list_result[result_type]['Section class']
1591 + self.result_effective_area = list_result[result_type]['Effective area']
1592
@@ -1732,125 +1752,134 @@ def design Strut(self):
1732 1752     # 2 - Based on optimum cost
1733 1753     self.optimum_section_cost_results = {}
1734 1754     self.optimum_section_cost = []
1755 + self.flag = self.section_classification(self)
1735 1756
1736 - if self.effective_area_factor == 1.0:
1757 + print('self.flag:',self.flag)
1758 + if self.effective_area_factor < 1.0:
1759 +     logger.warning(
1760 +         "Reducing the effective sectional area as per the definition in the Design Preferences tab."
1761 +     )

```

Figure 4.28: From compression.py


```

src/osdag/design_type/compression_member/compression.py
1874 +         print("Section result: \n",self.sec_profile, list_result)
1875 +         # Step 3 - Storing the optimum results to a list in a descending order
1876 +
1877 +         list_1 = ['Designation','Section class', 'Effective area', 'Buckling_class', 'IF',
1878 +                 'Effective_length', 'Effective_SR', 'EBS', 'lambda_vv', 'lambda_psi', 'ND_ESR', 'phi', 'SRF',
1879 +                 'FCD_formula', 'FCD_max', 'FCD', 'Capacity', 'UR', 'Cost']
1880 +         self.common_checks_1(self, section, 5, list_result, list_1)
1881 +         print(f"\n self.optimum_section_cost_results {self.optimum_section_cost_results}")
1882 +         f"\n self.optimum_section_ur_results {self.optimum_section_ur_results}")
1854 1883
1855 1884         def min_rad_gyration_calc_strut(self,designation, material_grade,key,subkey, D_a=0.0,B_b=0.0,T_t=0.0,t=0.0):
1856 1885             if key == Profile_name_1 and (subkey == loc_type1 or subkey == loc_type2):
@@ -2020,6 +2049,16 @@ def strength_of_strut(self):
2020 2049
2021 2050         def results(self):
2022 2051             """ """
2052 +             _ = [i for i in self.optimum_section_ur if i > 1.0]
2053 +             print( '_ ',_)
2054 +             if len(_)==1:
2055 +                 temp = _[0]
2056 +             elif len(_)==0:
2057 +                 temp = None
2058 +             else:
2059 +                 temp = sorted(_)[0]
2060 +             self.failed_design_dict = self.optimum_section_ur_results[temp] if temp is not None else None
2061 +             print('self.failed_design_dict ',self.failed_design_dict)
2023 2062             # sorting results from the dataset
2024 2063             #if len(self.input_section_list) > 1 : #TODO: Parth to confirm if this code is needed
2025 2064             #if design_dictionary[KEY_AXIAL] != '': #TODO: Parth to confirm if this code is needed
@@ -2039,29 +2078,44 @@ def results(self):
2039 2078             logger.error("The solver did not find any adequate section from the defined list.")
2040 2079             logger.info("Re-define the list of sections or check the Design Preferences option and re-design.")
2041 2080             self.design_status = False
2042 -             self.design_status_list.append(self.design_status)
2081 +             if len(self.failed_design_dict)>0:
2082 +                 logger.info(
2083 +                     "The details for the best section provided is being shown"
2084 +                 )
2085 +                 self.result_UR = self.failed_design_dict['UR'] #temp
2086 +                 self.common_result(
2087 +                     self,
2088 +                     list_result=self.failed_design_dict,
2089 +                     result_type=None,
2090 +                 )
2091 +                 logger.warning(
2092 +                     "Re-define the list of sections or check the Design Preferences option and re-design."
2093 +                 )
2094 +                 return
2095 +                 #self.design_status_list.append(self.design_status)
2043 2096
2044 2097             else:
2045 2098                 self.result_UR = self.optimum_section_ur[1] # optimum section which passes the UR check

```

Figure 4.29: From compression.py

```

src/osdag/design_type/compression_member/compression.py
2114 2141
@@ -2118,7 +2145,7 @@ def results(self, design_dictionary):
2118 2145     """ start writing save_design from here!
2119 2146     def save_design(self, popup_summary):
2120 2147
2121 -         if self.connectivity == 'Hollow/Tubular Column Base':
2148 +         """if self.connectivity == 'Hollow/Tubular Column Base':
2122 2149             if self.dp_column_designation[1:4] == 'SHS':
2123 2150                 select_section_img = 'SHS'
2124 2151             elif self.dp_column_designation[1:4] == 'RHS':
@@ -2140,23 +2167,51 @@ def save_design(self, popup_summary):
2140 2167                 else:
2141 2168                     section_type = 'Circular Hollow Section (CHS)'
2142 2169                 else:
2143 -                     section_type = 'I Section'
2170 +                     section_type = 'I Section' """
2171 +
2172 +                 if self.section_property.max_leg == self.section_property.min_leg:
2173 +                     if self.sec_profile == "Back to Back Angles":
2174 +                         if self.loc == "Long Leg":
2175 +                             image = "bblequaldp"
2176 +                         else:
2177 +                             image = "bbsequaldp"
2178 +                     elif self.sec_profile == "Star Angles":
2179 +                         if self.loc == "Long Leg":
2180 +                             image = "salequaldp"
2181 +                         else:
2182 +                             image = "sasequaldp"
2183 +                     else:
2184 +                         image = "equaldp"
2144 2185
2186 +                 else:
2187 +                     if self.sec_profile == "Back to Back Angles":
2188 +                         if self.loc == "Long Leg":
2189 +                             image = "bblunequaldp"
2190 +                         else:
2191 +                             image = "bbsunequaldp"
2192 +                     elif self.sec_profile == "Star Angles":
2193 +                         if self.loc == "Long Leg":
2194 +                             image = "salunequaldp"
2195 +                         else:
2196 +                             image = "sasunequaldp"
2197 +                     else:
2198 +                         image = "unequaldp"
2199 +
2145 2200
2146 -                 if self.section_property=='Columns' or self.section_property=='Beams':
2147 -                     self.report_column = {KEY_DISP_SEC_PROFILE: "ISection",
2148 -                                         KEY_DISP_COLSEC_REPORT: self.section_property.designation,

```

Figure 4.30: From compression.py

```

src/osdag/design_type/compression_member/compression.py
2095 + #self.design_status_list.append(self.design_status)
2043 2096
2044 2097
2045 - else:
2098 + self.result_UR = self.optimum_section_ur[-1] # optimum section which passes the UR check
2099 + self.failed_design_dict = None
2100 + self.result_UR = self.optimum_section_ur[
2101 + -1
2102 + ] # optimum section which passes the UR check
2046 2102 print(f"self.result_UR{self.result_UR}")
2047 2103 self.design_status = True
2104 + self.common_result(
2105 + self,
2106 + list_result=self.optimum_section_ur_results,
2107 + result_type=self.result_UR,
2108 + )
2048 2109
2049 2110 else: # results based on cost
2050 2111 self.optimum_section_cost.sort()
2051 2112
2052 2113 # selecting the section with most optimum cost
2053 2114 self.result_cost = self.optimum_section_cost[0]
2054 2115
2055 - # print results
2056 2116 if len(self.optimum_section_ur) == 0:
2057 - logger.warning(
2058 - "The sections selected by the solver from the defined list of sections did not satisfy the Utilization Ratio (UR) "
2059 - "criteria")
2060 - logger.error("The solver did not find any adequate section from the defined list.")
2061 - logger.info("Re-define the list of sections or check the Design Preferences option and re-design.")
2062 2117 self.design_status = False
2063 - self.design_status_list.append(self.design_status)
2064 - pass
2118 +
2065 2119 else:
2066 2120 if self.optimization_parameter == 'Utilization Ratio':
2067 2121 print(f" self.optimum_section_ur_results {self.optimum_section_ur_results}")
@@ -2198,154 +2252,177 @@ def save_design(self, popup_summary):
2198 2252 else:
2199 2253 image = "unequaldp"
2200 2254
2255 + if (self.design_status and self.failed_design_dict is None) or (not self.design_status and len(self.failed_design_dict)>0):
2256 + if self.sec_profile == "Angles" or self.sec_profile == VALUES_SEC_PROFILE_2[0]:
2257 + self.report_column = {KEY_DISP_SEC_PROFILE: image,
2258 + KEY_DISP_SECSIZE: (self.section_property.designation, self.sec_profile),
2259 + KEY_DISP_MATERIAL: self.section_property.material,
2260 + # KEY_DISP_APPLIED_AXIAL_FORCE: self.section_property.,
2261 + KEY_REPORT_MASS: self.section_property.mass,
2262 + KEY_REPORT_AREA: round(self.section_property.area * 1e-2, 2),
2263 + KEY_REPORT_MAX_LEG_SIZE: round(self.section_property.max_leg,2),
2264 + KEY_REPORT_MIN_LEG_SIZE: round(self.section_property.min_leg,2),
2265 + KEY_REPORT_ANGLE_THK: round(self.section_property.thickness,2),
2266 + KEY_REPORT_R1: self.section_property.root_radius,

```

Figure 4.31: From compression.py

```

src/osdag/design_type/compression_member/compression.py
2289 +         self.report_input = \
2290 +             {#KEY_MAIN_MODULE: self.mainmodule,
2291 +              KEY_MODULE: self.module, #"Axial load on column "
2292 +              KEY_DISP_AXIAL: self.load.axial_force/1000,
2293 +              KEY_DISP_LENGTH: self.length,
2294 +              KEY_DISP_SEC_PROFILE: self.sec_profile,
2295 +              KEY_DISP_END1: self.end_1,
2296 +              KEY_DISP_END2: self.end_2,
2297 +              KEY_DISP_SECSIZE: self.result_section_class,
2298 +              "Strut Section - Mechanical Properties": "TITLE",
2299 +              KEY_DISP_ULTIMATE_STRENGTH_REPORT: round(self.section_property.fu, 2),
2300 +              KEY_DISP_YIELD_STRENGTH_REPORT: round(self.section_property.fy, 2),
2301 +              KEY_MATERIAL: self.material,
2302 +              KEY_DISP_EFFECTIVE_AREA_PARA: self.effective_area_factor,
2303 +              KEY_DISP_SECSIZE: str(self.sec_list),
2304 +              "Selected Section Details": self.report_column,
2305 +             }
2306 +
2307 +         self.report_check = []
2308 +
2309 +         t1 = ('Selected', 'Selected Member Data', '|p{5cm}|p{2cm}|p{2cm}|p{4cm}|')
2310 +         self.report_check.append(t1)
2311 +
2312 +
2313 +         t1 = ('SubSection', 'Buckling Class & Imperfection factor', '|p{4cm}|p{2 cm}|p{7cm}|p{3 cm}|')
2314 +         self.report_check.append(t1)
2315 +         t1 = (KEY_DISP_BUCKLING_CURVE_ZZ, ' ',
2316 +              cl_8_7_1_5_buckling_curve(),
2317 +              ' ')
2318 +         self.report_check.append(t1)
2319 +
2320 +         t1 = (KEY_DISP_IMPERFECTION_FACTOR_ZZ + r' ($\alpha$)', ' ',
2321 +              cl_8_7_1_5_imperfection_factor(self.result_IF),
2322 +              ' ')
2323 +         self.report_check.append(t1)
2324 +
2325 +         t1 = ('SubSection', 'Section Classification', '|p{5cm}|p{3cm}|p{6.5cm}|p{1.5cm}|')
2326 +         self.report_check.append(t1)
2327 +         self.h = (self.section_property.leg_a_length - 2 * (self.section_property.thickness + self.section_property.root_radius))
2328 +         t1 = ('Single Angle',
2329 +              cl_3_7_2_section_classification_angle_required("b/t", self.section_property.section_class),
2330 +              cl_3_7_2_section_classification_angle_provided(
2331 +                  self.section_property.min_leg, self.section_property.max_leg, self.section_property.thickness,
2332 +                  round(self.width_thickness_ratio, 2), "b/t", self.epsilon, self.section_property.section_class),
2333 +              get_pass_fail(15.7 * self.epsilon, round(self.width_thickness_ratio, 2), relation="geq")
2334 +              )
2335 +         self.report_check.append(t1)
2336 +
2337 +         t1 = (
2338 +             'Double Angles with the components separated',
2339 +             cl_3_7_2_section_classification_angle_required("d/t", self.section_property.section_class),
2340 +             cl_3_7_2_section_classification_angle_provided(
2341 +                 self.section_property.min_leg, self.section_property.max_leg, self.section_property.thickness,

```

Figure 4.32: From compression.py

```

src/osdag/design_type/compression_member/compression.py
2337 +         t1 = (
2338 +             'Double Angles with the components separated',
2339 +             c1_3_7_2_section_classification_angle_required("d/t", self.section_property.section_class),
2340 +             c1_3_7_2_section_classification_angle_provided(
2341 +                 self.section_property.min_leg, self.section_property.max_leg, self.section_property.thickness,
2342 +                 round(self.depth_thickness_ratio, 2), "d/t", self.epsilon, self.section_property.section_class),
2343 +             get_pass_fail(15.7 * self.epsilon, round(self.depth_thickness_ratio, 2), relation="geq")
2344 +         )
2345 +         self.report_check.append(t1)
2346 +
2347 +         t1 = (
2348 +             'Axial Compression',
2349 +             c1_3_7_2_section_classification_angle_required("(b+d)/t", self.section_property.section_class),
2350 +             c1_3_7_2_section_classification_angle_provided(
2351 +                 self.section_property.min_leg, self.section_property.max_leg, self.section_property.thickness,
2352 +                 round(self.width_depth_thickness_ratio, 2), "(b+d)/t", self.epsilon, self.section_property.section_class),
2353 +             get_pass_fail(25 * self.epsilon, round(self.width_depth_thickness_ratio, 2), relation="geq")
2354 +         )
2355 +         self.report_check.append(t1)
2234 2356
2235 -         self.report_input = \
2357 +         t1 = ('(All the above three criteria should be satisfied)', '', '', '')
2358 +         self.report_check.append(t1)
2359 +
2360 +
2361 +         t1 = ('Section Class', '',
2362 +             c1_3_7_2_section_classification(
2363 +                 self.section_property.section_class),
2364 +             '')
2365 +         self.report_check.append(t1)
2366 +
2367 +         t1 = ('SubSection', 'Effective Slenderness Ratio', '|p{4cm}|p{2 cm}|p{7cm}|p{3 cm}|')
2368 +         self.report_check.append(t1)
2369 +         if self.load_type == 'Concentric Load':
2370 +             K = self.result_eff_len / self.length
2371 +             t1 = ("Effective Slenderness Ratio", '',
2372 +                 c1_7_1_2_effective_slenderness_ratio(K, self.length, round(self.min_radius_gyration, 2), self.result_eff_sr),
2373 +                 '')
2374 +             self.report_check.append(t1)
2375 +         else:
2376 +             t1 = ("Effective Slenderness Ratio", '',
2377 +                 c1_7_5_1_2_effective_slenderness_ratio(self.k1, self.k2, self.k3, self.lambda_vv, self.lambda_psi, self.result_eff_sr),
2378 +                 '')
2379 +             self.report_check.append(t1)
2380 +
2381 +
2382 +         t1 = ('SubSection', 'Checks for Strength', '|p{4cm}|p{2 cm}|p{7cm}|p{3 cm}|')
2383 +         self.report_check.append(t1)
2384 +         t1 = (KEY_DISP_EULER_BUCKLING_STRESS_ZZ, '',
2385 +             c1_8_7_1_5_buckling_stress(self.section_property.modulus_of_elasticity, self.result_eff_sr, round(self.result_ebs, 2)),
2386 +             symbol_to_lanbdae, '')
2387 +         self.report_check.append(t1)

```

Figure 4.33: From compression.py

```

2393 +         t1 = (r'$F_{cd} \le \left( \frac{N}{A} \right)_{\text{right}}$', '',
2394 +             c1_8_7_1_5_Buckling(self.material_property.fy, self.gamma_m0, round(self.nondimensional_effective_slenderness_ratio, 2), round(self.phi,
2395 +             2), round(self.design_compressive_stress_max, 2), round(self.design_compressive_stress, 2)), '')
2396 +         self.report_check.append(t1)
2397 +
2398 +         t1 = ('P_d', self.load.axial_force * 10 ** -3,
2399 +             c1_7_1_2_design_compressive_strength(round(self.result_capacity * 10 ** -3, 2), round(self.result_effective_area, 2),
2400 +             round(self.design_compressive_stress, 2), self.load.axial_force * 10 ** -3),
2401 +             get_pass_fail(self.load.axial_force * 10 ** -3, round(self.result_capacity * 10 ** -3, 2), relation="leq"))
2402 +         self.report_check.append(t1)
2403 +
2404 +         else:
2236 2405             self.report_input = \
2237 2406                 {#KEY_MAIN_MODULE: self.mainmodule,
2238 2407                 KEY_MODULE: self.module, #Axial load on column "
2239 2408                 KEY_DISP_AXIAL: self.load.axial_force/1000,
2240 2409                 KEY_DISP_LENGTH: self.length,
2241 2410                 KEY_DISP_SEC_PROFILE: self.sec_profile,
2242 2411                 KEY_DISP_END1: self.end_1,
2243 -                 KEY_DISP_SECSIZE: self.result_section_class,
2412 +                 #KEY_DISP_SECSIZE: self.result_section_class,
2244 2413                 "Strut Section - Mechanical Properties": "TITLE",
2245 2414                 KEY_DISP_ULTIMATE_STRENGTH_REPORT: round(self.section_property.fu, 2),
2246 2415                 KEY_DISP_YIELD_STRENGTH_REPORT: round(self.section_property.fy, 2),
2247 2416                 KEY_MATERIAL: self.material,
2248 2417                 KEY_DISP_EFFECTIVE_AREA_PARA: self.effective_area_factor,
2249 2418                 KEY_DISP_SECSIZE: str(self.sec_list),
2250 -                 "Selected Section Details": self.report_column,
2251 -                 }
2252 -

```

Figure 4.34: From compression.py

```

2348 +
2349 +     Disp_2d_image = []
2350 +     Disp_3D_image = "/ResourceFiles/images/3d.png"
2349 2351     print(sys.path[0])
2350 2352     rel_path = str(sys.path[0])
2351 2353     rel_path = os.path.abspath(".") # TEMP
2352 2354     rel_path = rel_path.replace("\\", "/")
2353 2355     fname_no_ext = popup_summary['filename']
2354 2356     CreateLatex.save_latex(CreateLatex(), self.report_input, self.report_check, popup_summary, fname_no_ext,
2355 -                             rel_path, [], '', module=self.module)
2357 +                             rel_path, Disp_2d_image, Disp_3D_image, module=self.module)
2356 2358
2357 2359
2358 2360     # def memb_pattern(self, status):
@@ -2388,4 +2390,5 @@ def save_design(self, popup_summary):
2388 2390     #     ['./ResourceFiles/images/L.png',400,202, "Plate Block Shear Pattern"]) # [image, width, height, caption]
2389 2391     #     pattern.append(t99)
2390 2392     #
2391 -     #     return pattern
2393 +     #     return pattern
2394 +

```

Figure 4.35: From compression.py

Chapter 5

Internship Task 3 Title: Miscellaneous

5.1 Task 3: Problem Statement

Several issues and errors were encountered across various functionalities and features, including image rendering problems, unexpected float type errors, and ineffective input/output docks. Specific challenges were noted in the BC-End Plate Typical Sketch and the compression members' output dock, which failed to perform as intended. Additionally, discrepancies in selected member data and ineffective lengths for flexural members further complicated the workflow, requiring corrections and updates to ensure accuracy and functionality.

5.2 Task 3: Tasks Done

The image issues and float type errors were resolved by type-casting the problem causing variables to `int()`. Adjustments were made to the input dock image paths by converting them to absolute path instead of relative, ensuring seamless operation and addressing any image-related discrepancies. The BC-End Plate Typical Sketch was causing an error due to a similar float-type error in the `ui_template.py` file and was rectified again by type-casting the problem causing variables to `int()` to reflect the required design sketch correctly. The output dock for compression members and the selected member data functionality were fixed to provide accurate results of the most optimum section instead of a random section from the input list as it was doing before. Effective lengths for flexural member modules were fixed to show the lengths for the most optimum section

instead of a random section, which was varying each time to ensure compliance with design standards. These fixes collectively enhanced the reliability and precision of the system.

5.3 Task 3: Python Code

The bug-fixes for some of these miscellaneous tasks are described below:

5.3.1 For fixing Output Dock Designation and Selected Member Data

```

src/osdag/design_type/compression_member/Column.py
895 - #logger.info("The actual effective area is {} mm2 and the reduced effective area is {} mm2 [Reference: Cl. 7.3.2, IS 800:2007]".
896 - #         format(round((self.effective_area / self.effective_area_factor), 2), self.effective_area))
897 - #else:
898 - #if self.section_class != 'Slender':
899 - #     logger.info("The effective sectional area is taken as 100% of the cross-sectional area [Reference: Cl. 7.3.2, IS 800:2007].")
900 890
901 891         #print('self.input_section_list:',self.input_section_list)
902 892         if self.flag:
903 +
904 @@ -923,7 +913,7 @@ def design_column(self):
923 913
924 914             self.material_property.connect_to_database_to_get_fy_fu(self.material, max(self.section_property.flange_thickness,
925 915             self.section_property.web_thickness))
926 -
927 916 +             self.epsilon = math.sqrt(250 / self.material_property.fy)
927 917
928 918
929 919             # initialize lists for updating the results dictionary
930 +
931 @@ -937,9 +927,6 @@ def design_column(self):
937 927             self.section_class = self.input_section_classification[section][0]
938 928
939 929             if self.section_class == 'Slender':
940 - #logger.warning("The trial section ({} is Slender. Computing the Effective Sectional Area as per Sec. 9.7.2, "
941 - #                 "Fig. 2 (B & C) of The National Building Code of India (NBC), 2016.".format(section))
942 -
943 930             if (self.sec_profile == VALUES_SEC_PROFILE[0]): # Beams and Columns
944 931                 self.effective_area = (2 * ((31.4 * self.epsilon * self.section_property.flange_thickness) *
945 932                 self.section_property.flange_thickness)) + \
946 +
947 @@ -950,6 +937,9 @@ def design_column(self):
950 937             self.effective_area = self.section_property.area # mm2
951 938             # print(f"self.effective_area{self.effective_area}")
952 939
953 940 +             if self.effective_area_factor < 1.0:
954 941 +                 self.effective_area = round(self.effective_area * self.effective_area_factor, 2)
955 942 +
956 943             self.list_zz.append(self.section_class)
957 944             self.list_yy.append(self.section_class)
958 945
959 +
960 @@ -1416,6 +1406,11 @@ def save_design(self, popup_summary):
1416 1406
1417 1407             if (self.design_status and self.failed_design_dict is None) or (not self.design_status and len(self.failed_design_dict)>0):
1418 1408                 if self.sec_profile=='Columns' or self.sec_profile=='Beams' or self.sec_profile == VALUES_SEC_PROFILE[0]:
1419 1409 +                     try:
1420 1410 +                         result = Beam(designation=self.result_designation, material_grade=self.material)
1421 1411 +                     except:
1422 1412 +                         result = Column(designation=self.result_designation, material_grade=self.material)
1423 1413 +                 self.section_property = result
1424 1414                 self.report_column = {KEY_DISP_SEC_PROFILE: "ISection",
1425 1415                 KEY_DISP_SECSIZE: (self.section_property.designation, self.sec_profile),
1426 1416                 KEY_DISP_COLSEC_REPORT: self.section_property.designation,

```

Figure 5.1: From Column.py


```

src/osdag/design_type/compression_member/compression.py
2255 + if self.sec_profile == Profile_name_1 or self.sec_profile == Profile_name_2 or self.sec_profile == Profile_name_3: # Angles and Back to Back Angles
2257 + self.section_property = Angle(designation = self.result_designation, material_grade = self.material)
2256 2258 if self.sec_profile == "Angles" or self.sec_profile == VALUES_SEC_PROFILE_2[0]:
2257 2259 self.report_column = {KEY_DISP_SEC_PROFILE: image,
2258 2260 KEY_DISP_SECSIZE: (self.section_property.designation, self.sec_profile),
@@ -2326,30 +2328,30 @@ def save_design(self, popup_summary):
2326 2328 self.report_check.append(t1)
2327 2329 self.h = (self.section_property.leg_a_length - 2 * (self.section_property.thickness + self.section_property.root_radius))
2328 2330 t1 = ('Single Angle',
2329 - c1_3_7_2_section_classification_angle_required("b/t", self.section_property.section_class),
2331 + c1_3_7_2_section_classification_angle_required("b/t", self.input_section_classification[self.result_designation]),
2330 2332 c1_3_7_2_section_classification_angle_provided(
2331 2333 self.section_property.min_leg, self.section_property.max_leg, self.section_property.thickness,
2332 - round(self.width_thickness_ratio, 2), "b/t", self.epsilon, self.section_property.section_class),
2334 + round(self.width_thickness_ratio, 2), "b/t", self.epsilon, self.input_section_classification[self.result_designation]),
2333 2335 get_pass_fail(15.7 * self.epsilon, round(self.width_thickness_ratio, 2), relation="geq")
2334 2336 )
2335 2337 self.report_check.append(t1)
2336 2338
2337 2339 t1 = (
2338 2340 'Double Angles with the components separated',
2339 - c1_3_7_2_section_classification_angle_required("d/t", self.section_property.section_class),
2341 + c1_3_7_2_section_classification_angle_required("d/t", self.input_section_classification[self.result_designation]),
2340 2342 c1_3_7_2_section_classification_angle_provided(
2341 2343 self.section_property.min_leg, self.section_property.max_leg, self.section_property.thickness,
2342 - round(self.depth_thickness_ratio, 2), "d/t", self.epsilon, self.section_property.section_class),
2344 + round(self.depth_thickness_ratio, 2), "d/t", self.epsilon, self.input_section_classification[self.result_designation]),
2343 2345 get_pass_fail(15.7 * self.epsilon, round(self.depth_thickness_ratio, 2), relation="geq")
2344 2346 )
2345 2347 self.report_check.append(t1)
2346 2348
2347 2349 t1 = (
2348 2350 'Axial Compression',
2349 - c1_3_7_2_section_classification_angle_required("(b+d)/t", self.section_property.section_class),
2351 + c1_3_7_2_section_classification_angle_required("(b+d)/t", self.input_section_classification[self.result_designation]),
2350 2352 c1_3_7_2_section_classification_angle_provided(
2351 2353 self.section_property.min_leg, self.section_property.max_leg, self.section_property.thickness,
2352 - round(self.width_depth_thickness_ratio, 2), "(b+d)/t", self.epsilon, self.section_property.section_class),
2354 + round(self.width_depth_thickness_ratio, 2), "(b+d)/t", self.epsilon, self.input_section_classification[self.result_designation]),
2353 2355 get_pass_fail(25 * self.epsilon, round(self.width_depth_thickness_ratio, 2), relation="geq")
2354 2356 )
2355 2357 self.report_check.append(t1)
@@ -2359,9 +2361,7 @@ def save_design(self, popup_summary):
2359 2361
2360 2362 t1 = ('Section Class', ' ',
2361 2363 c1_3_7_2_section_classification(
2362 - self.section_property.section_class),
2363 - ' ')
2364 + c1_3_7_2_section_classification(self.input_section_classification[self.result_designation]), ' ')
2365 2365 self.report_check.append(t1)

```

Figure 5.2: From compression.py

5.3.2 For fixing Input and Output Dock Images and Sketch

These path changes were made to all the python source files for each module of the Connection and Compression Members, wherever suitable.

```

src/osdag/design_type/connection/beam_beam_end_plate_splice.py
@@ -37,6 +37,7 @@
37 37 from ...design_report.reportGenerator import save_html
38 38 from ...Report_functions import *
39 39 from ...design_report.reportGenerator_latex import CreateLatex
40 +from importlib.resources import files
40 41
41 42 import logging
42 43 import math

@@ -224,7 +225,7 @@ def input_values(self):
224 225     t2 = (KEY_ENDPLATE_TYPE, KEY_DISP_ENDPLATE_TYPE, TYPE_COMBOBOX, VALUES_ENDPLATE_TYPE, True, 'No Validator')
225 226     options_list.append(t2)
226 227
227 -     t15 = (KEY_IMAGE, None, TYPE_IMAGE, './ResourceFiles/images/flush_ep.png', True, 'No Validator')
228 +     t15 = (KEY_IMAGE, None, TYPE_IMAGE, str(files("osdag.data.ResourceFiles.images").joinpath("flush_ep.png")), True, 'No Validator')
228 229     options_list.append(t15)
229 230
230 231     t4 = (KEY_SUPTDSEC, KEY_DISP_BEAMSEC, TYPE_COMBOBOX, connectdb("Beams"), True, 'No Validator')

@@ -284,19 +285,21 @@ def input_value_changed(self):
284 285
285 286     return lst
286 287
288 +
287 289     def fn_conn_image(self):
288 -         """ display representative images of end plate type """
289 +         """ Display representative images of end plate type """
289 291
290 291     ep_type = self[0]
290 292     if ep_type == VALUES_ENDPLATE_TYPE[0]:
291 293         return './ResourceFiles/images/flush_ep.png'
292 -         return str(files("osdag.data.ResourceFiles.images").joinpath("flush_ep.png"))
293 +         elif ep_type == VALUES_ENDPLATE_TYPE[1]:
294 -         return './ResourceFiles/images/owe_ep.png'
295 +         return str(files("osdag.data.ResourceFiles.images").joinpath("owe_ep.png"))
296 +         elif ep_type == VALUES_ENDPLATE_TYPE[2]:
297 -         return './ResourceFiles/images/extended.png'
298 +         return str(files("osdag.data.ResourceFiles.images").joinpath("extended.png"))
299 299     else:
298 300         return ''
299 301
302 +
300 303     # create customized input for UI
301 304     def customized_input(self):
302 305         """ list of values available with customize option"""

@@ -483,15 +486,15 @@ def stiffener_detailing(self, status):
483 486     detailing = []
484 487
485 488     if self.endplate_type == VALUES_ENDPLATE_TYPE[0]: # Flush EP
486 -         detailing_path = './ResourceFiles/images/BB_Stiffener_FP.png'
489 +         detailing_path = str(files("osdag.data.ResourceFiles.images").joinpath("BB_Stiffener_FP.png"))

```

Figure 5.3: From compression.py

```

src/osdag/gui/ui_template.py
@@ -2273,7 +2273,7 @@ def output_button_dialog(self, main, button_list, button):
2273 2273
2274 2274         if value is not None and value != "":
2275 2275             im = QtWidgets.QLabel(image_widget)
2276 -             im.setFixedSize(value[1], value[2])
2276 +             im.setFixedSize(int(value[1]), int(value[2]))
2277 2277             pmap = QPixmap(value[0])
2278 2278             im.setScaledContents(1)
2279 2279             im.setStyleSheet("background-color: white;")
@@ -2282,7 +2282,7 @@ def output_button_dialog(self, main, button_list, button):
2282 2282             caption = QtWidgets.QLabel(image_widget)
2283 2283             caption.setAlignment(Qt.AlignCenter)
2284 2284             caption.setText(value[3])
2285 -             caption.setFixedSize(value[1], caption.sizeHint().height())
2285 +             caption.setFixedSize(int(value[1]), caption.sizeHint().height())
2286 2286             image_layout.addWidget(caption)
2287 2287             max_image_width = max(max_image_width, value[1])
2288 2288             max_image_height = max(max_image_height, value[2])
@@ -2312,15 +2312,15 @@ def output_button_dialog(self, main, button_list, button):
2312 2312             if option_type == TYPE_IMAGE:
2313 2313                 im = QtWidgets.QLabel(image_widget)
2314 2314                 im.setScaledContents(True)
2315 -                 im.setFixedSize(value[1], value[2])
2315 +                 im.setFixedSize(int(value[1]), int(value[2]))
2316 2316                 pmap = QPixmap(value[0])
2317 2317                 im.setStyleSheet("background-color: white;")
2318 2318                 im.setPixmap(pmap)
2319 2319                 image_layout.addWidget(im)
2320 2320                 caption = QtWidgets.QLabel(image_widget)
2321 2321                 caption.setAlignment(Qt.AlignCenter)
2322 2322                 caption.setText(value[3])
2323 -                 caption.setFixedSize(value[1], 12)
2323 +                 caption.setFixedSize(int(value[1]), 12)
2324 2324                 image_layout.addWidget(caption)
2325 2325                 max_image_width = max(max_image_width, value[1])
2326 2326                 max_image_height = max(max_image_height, value[2])
@@ -2344,8 +2344,8 @@ def output_button_dialog(self, main, button_list, button):
2344 2344             dialog_height = max(dialog_height, max_image_height+125)
2345 2345             if not no_note:
2346 2346                 dialog_height += 40
2347 -                 dialog.resize(dialog_width, dialog_height)
2348 -                 dialog.setMinimumSize(dialog_width, dialog_height)
2347 +                 dialog.resize(int(dialog_width), int(dialog_height))
2348 +                 dialog.setMinimumSize(int(dialog_width), int(dialog_height))
2349 2349
2350 2350             if no_note:
2351 2351                 layout1.removeWidget(note_widget)

```

Figure 5.4: From compression.py

Chapter 6

Internship Task 4 Title: Documentation of Report Generation and the New Installer Method

6.1 Task 3: Problem Statement

Effective documentation is required to support the ongoing development and maintenance of Osdag, focusing on two critical areas: the report generation process and the new installer method. For report generation, there is a need to document how reports are created, including the commands, scripts, and files involved, to provide a clear reference for new interns and project staff. Additionally, the philosophy and implementation of the new installer method need to be articulated to ensure a comprehensive understanding of its purpose and functionality, enabling seamless onboarding and troubleshooting.

6.2 Task 3: Tasks Done

The report generation process in Osdag has been meticulously documented, outlining the sequence of steps, key files, and commands utilized. This includes details about the specific scripts that control report formatting, how each data is processed, and output generation, providing a practical guide for newcomers. For the new installer method, the underlying philosophy and its implementation steps have been clearly articulated. This documentation highlights the rationale behind the new approach, the changes made from

previous methods, and the specific functionalities added. Together, these efforts create a robust resource to assist future contributors in understanding and improving these areas.

6.3 Task 3: Documentation

The links for documentation are as follows:

- For Report Generation: [Click here](#).
- For New Installer Method: [Click here](#).

Chapter 7

Conclusions

7.1 Tasks Accomplished

The issues related to the database, themes, and image paths have been successfully addressed for the new installer method. Additionally, report generation has been completed for the "Beams and Column" section profile in the "Column with Known Support Conditions" module, as well as for the "Angles" section profile in the "Struts in Trusses" module. The issues regarding co-relation of CAD generated with the optimum section results has been fixed, and the design log and output dock functionality have been fully resolved for all section profiles across both modules. Comprehensive documentation has been prepared detailing the report generation process and the new installer method.

7.2 Skills Developed

Throughout the internship, I strengthened my technical capabilities and collaborative skillset. I gained more knowledge about GitHub's advanced features for version control, including branch management, pull request workflows, and merge conflict resolution. Using LaTeX, I created comprehensive technical documentation and reports while adhering to professional formatting standards. I significantly expanded my Python programming expertise by implementing the use of different libraries, and optimizing code performance. In the area of software development practices, I gained hands-on experience with code review methodologies, receiving constructive feedback from my mentors while maintaining code quality standards. Through working with Conda, I developed proficiency in man-

aging virtual environments, handling package dependencies, and building distributable applications. The collaborative nature of the internship enhanced my ability to work effectively in cross-functional teams, communicate technical concepts clearly, and coordinate multiple project timelines simultaneously.

Chapter A

Appendix

A.1 Work Reports

| Sr. No. | Date | Day | Task | Hours Spent |
|---------|------------------|-----------|--|---|
| 1 | 12 November 2024 | Tuesday | Joining Install instructions for Osdag Session with Ajinkya to discuss new intended process and the current stage of the application. | 2 to 3 hours. |
| 2 | 13 November 2024 | Wednesday | Attempt to install on personal devices and record behaviour. | 4 to 5 hours. (Kept running into issues.) |
| 3 | 14 November 2024 | Thursday | Meeting with Ajinkya to finalize schedule. | 1 hour. |
| 4 | 15 November 2024 | Friday | Attempt installing using a Windows VM, and try on Mac again. | 5 hours. (Found the correct process-flow to follow and successfully installed.) |
| 5 | 16 November 2024 | Saturday | Weekly Holiday Successfully installed and got the application working on both the machines. | |
| 6 | 17 November 2024 | Sunday | Weekly Holiday | |
| 7 | 18 November 2024 | Monday | Study break for examinations (Followed up with the progress of the team, tried debugging few errors in the report generation.) | Spent the time according to the breaks in between the exams. |
| 8 | 19 November 2024 | Tuesday | | |
| 9 | 20 November 2024 | Wednesday | | |
| 10 | 21 November 2024 | Thursday | | |
| 11 | 22 November 2024 | Friday | | |
| 12 | 23 November 2024 | Saturday | | |
| 13 | 24 November 2024 | Sunday | | |
| 14 | 25 November 2024 | Monday | | |
| 15 | 26 November 2024 | Tuesday | | |
| 16 | 27 November 2024 | Wednesday | | |
| 17 | 28 November 2024 | Thursday | Resolve few path issues See how temp image files (the last 4 images in the report generated) are created, tried implementing the tempfile module. | 4 hours. |
| 18 | 29 November 2024 | Friday | | 4 to 4 and a half hours. |
| 19 | 30 November 2024 | Saturday | Weekly Holiday | |
| 20 | 1 December 2024 | Sunday | Weekly Holiday Read about NSIS to familirize myself Tried tinkering with the existing Windows Installer script. | |
| 21 | 2 December 2024 | Monday | Tested all the modules of the application, resolved few unexpected float errors, and checked the PyQt-GUI, rendering and report generation of these modules. | 4 hours. (Took a bit of time in doing trial and error for the input specifications of the modules.) |
| 22 | 3 December 2024 | Tuesday | Travel | |
| 23 | 4 December 2024 | Wednesday | Follow up with Parth regarding the latex report generation of Compression Members. | |
| 24 | 5 December 2024 | Thursday | Worked on the Column with know support conditions LaTeX Report Generation and Integration with CAD. | Spent 6 to 7 hours as per work required. |
| 25 | 6 December 2024 | Friday | | Spent 6 to 7 hours as per work required. |
| 26 | 7 December 2024 | Saturday | | 3 to 4 hours. |
| 27 | 8 December 2024 | Sunday | | 3 to 4 hours. |
| 28 | 9 December 2024 | Monday | | Spent 6 to 7 hours as per work required. |
| 29 | 10 December 2024 | Tuesday | | Spent 6 to 7 hours as per work required. |
| 30 | 11 December 2024 | Wednesday | | Spent 6 to 7 hours as per work required. |
| 31 | 12 December 2024 | Thursday | | Spent 6 to 7 hours as per work required. |
| 32 | 13 December 2024 | Friday | | Spent 6 to 7 hours as per work required. |
| 33 | 14 December 2024 | Saturday | | 3 to 4 hours. |
| 34 | 15 December 2024 | Sunday | 3 to 4 hours. | |
| 35 | 16 December 2024 | Monday | Worked on the Struts in Trusses LaTeX Report Generation and Integration with CAD. Also spent time on working with the new installer method, and NSIS part. | Spent 4 to 5 hours as per work required. |
| 36 | 17 December 2024 | Tuesday | | Spent 4 to 5 hours as per work required. |
| 37 | 18 December 2024 | Wednesday | | Spent 4 to 5 hours as per work required. |
| 38 | 19 December 2024 | Thursday | | Spent 4 to 5 hours as per work required. |
| 39 | 20 December 2024 | Friday | Fixed the image issues related to Input Dock and resolved conflicts with merging. | 4 hours. |
| 40 | 21 December 2024 | Saturday | Weekly Holiday | |
| 41 | 22 December 2024 | Sunday | Weekly Holiday Work on Documentation for New Installer. | 2 hours. |
| 42 | 23 December 2024 | Monday | Resolved Output Dock and Design Log for Compression Members. | 4 hours. |
| 43 | 24 December 2024 | Tuesday | | 2 hours. |
| 44 | 25 December 2024 | Wednesday | Fixed Effective Length Issue in Flexure Members. | 3 hours. |
| 45 | 26 December 2024 | Thursday | Fixed Selected Member Data Issue in Compression Members. | 4 hours. |
| 46 | 27 December 2024 | Friday | | 3 hours. |
| 47 | 28 December 2024 | Saturday | Worked on Documentation for Report Generation and New Installer Method. | 5 hours. |

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.