FOSSEE Winter Internship Report

On

Development of LaTeX Environment for Report Generation in Life Cycle Cost Analysis(LCCA) Application for Osdag

Submitted by

Ishan Rai

3rd Year B.Tech Student, Department of

VIT Bhopal University

Under the Guidance of

Prof. Siddhartha Ghosh

Department of Civil Engineering

Indian Institute of Technology Bombay

Mentor:

Ajmal Babu M S

Parth Karia

Ajinkya Dahale

February 22, 2026

Acknowledgments

I would like to express my sincere gratitude to everyone who supported and guided me throughout my internship and project work. I am especially thankful to the Osdag team members Ajmal Babu M. S., Ajinkya Dahale, and Parth Karia for their valuable guidance and continuous support. I also extend my heartfelt thanks to Prof. Siddhartha Ghosh, Principal Investigator of Osdag, Department of Civil Engineering, IIT Bombay, for his leadership and encouragement.

I am deeply grateful to Prof. Kannan M. Moudgalya, Principal Investigator of the FOSSEE Project, Department of Chemical Engineering, IIT Bombay, and to FOSSEE Managers Ms. Usha Viswanathan and Ms. Vineeta Parmar for their constant support and coordination. I also acknowledge the support of the National Mission on Education through ICT, Ministry of Education, Government of India.

Finally, I thank my colleagues, my college, and the Placement Team(PAT) at VIT Bhopal for their encouragement and assistance throughout this journey.

Contents

Chapter 1

Introduction

1.1 National Mission in Education through ICT

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

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

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

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

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

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

1.1.1 ICT Initiatives of MoE

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

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

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

1.2 FOSSEE Project

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

1.2.1 Projects and Activities

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

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

1.2.2 Fellowships

FOSSEE offers various internship and fellowship opportunities for students:

- Winter Internship
- Summer Fellowship
- Semester-Long Internship

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

For more details, visit the official FOSSEE website.

Figure 1.1: FOSSEE Projects and Activities

1.3 Osdag Software

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

1.3.1 Osdag GUI

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

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

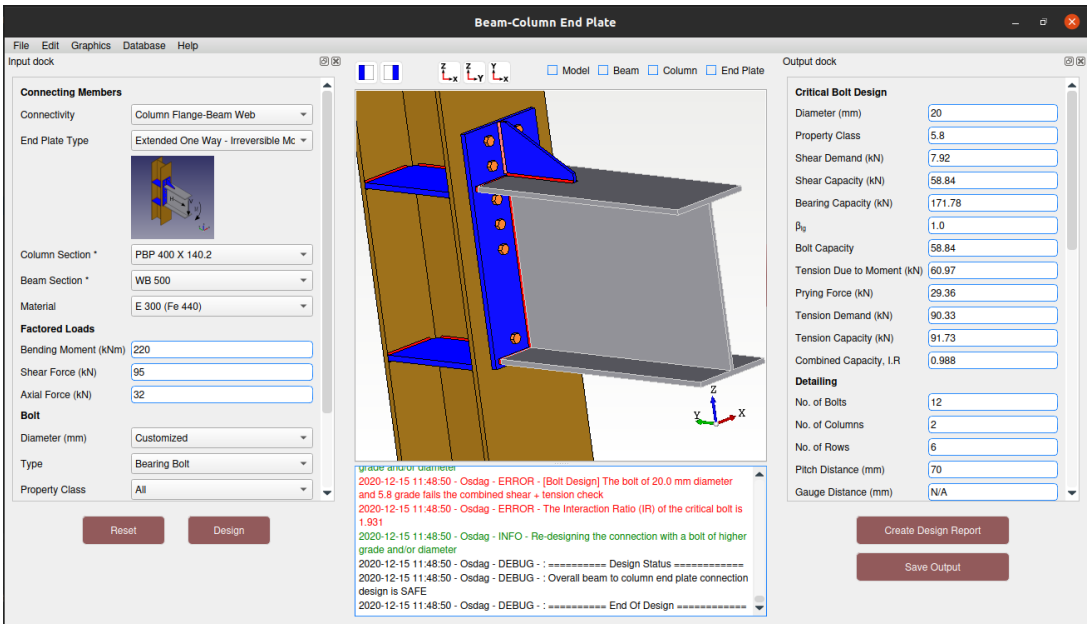


Figure 1.2: Osdag GUI

1.3.2 Features

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

For more details, visit the official Osdag website.

Chapter 2

Screening Task

2.1 Problem Statement

You are required to create a Python script that reads a force table from an Excel file and generates a PDF engineering report using PyLaTex. The report must include an introduction with an embedded beam image, a force table recreated as a LaTeX table (not an image), and Shear Force Diagram (SFD) and Bending Moment Diagram (BMD) plotted using TikZ/pgfplots (no image-based plots allowed).

2.2 Tasks Done

The following tasks were carried out as part of the Simply Supported Beam Analysis. The beam system was defined and analyzedto determine internal forces and structural behavior.

- Calculated support reactions using $\sum F_y = 0$ and $\sum M = 0$..
- Developed the Shear Force Diagram (SFD) and Bending Moment Diagram (BMD).
- Identified maximum shear (45 kN at supports) and maximum bending moment (168.75 kNm at 7.5 m).

All calculations were verified using linear elastic beam theory to ensure accuracy and reliability.

Chapter 3

Internship Task 1: Unit Testing of Osdag and LaTeX Report Template Development for OSBridgeLCCA

3.1 Task 1: Problem Statement

The first objective of this project was to perform unit testing on Osdag to ensure the correctness, stability, and proper functioning of its individual modules. This process involved verifying that each unit produced the expected output and identifying any issues affecting reliability.

In addition, the OSBridgeLCCS application was explored to understand its functionality, data structure, and reporting requirements. Based on this analysis, a LaTeX report template was designed and developed to serve as a standardized format for Life Cycle Cost Analysis reports. The template was created to ensure clear presentation, proper organization of data, and professional documentation, forming the foundation for automated report generation.

3.2 Task 1: Tasks Done

Task 1 consisted of two major components: mass unit testing of the Header Plate / End Plate Connection module in Osdag and development of a LaTeX-based financial report generator along with a report bridge for automated report generation in the OSBridgeL-

CCA application.

3.2.1 Mass Unit Testing of Header Plate / End Plate Connection Module

Methodology

A mass unit testing approach was adopted to verify the correctness and reliability of the Header Plate / End Plate Connection module in Osdag. The testing ensured that the module produced correct design outputs and generated proper reports under different input conditions.

Process

The module was studied to understand its workflow, including input parameters, design calculations, and report generation. Multiple test cases were prepared by varying parameters such as bolt diameter, plate thickness, and applied load. The module was executed repeatedly, and the generated reports were verified.

Implementation

Testing was performed using the Osdag graphical user interface. The connection design was executed for multiple input cases, and the generated design reports were verified for correctness and completeness.

Outcome

The testing confirmed that the Header Plate / End Plate Connection module was functioning correctly and generating accurate design reports.

3.2.2 Development of Financial Report Generator and Report Bridge

Methodology

A LaTeX-based financial report generation system was developed to automate the creation of Life Cycle Cost Analysis reports. The system consisted of a LaTeX report

template, a financial report generator script, and a report bridge to transfer input data from the OSBridgeLCCA application.

Process

A LaTeX template was created with placeholders for dynamic data insertion. A Python script called FinancialReportGenerator was developed to read the template, replace placeholders with actual values, and generate a PDF report.

A report bridge function was implemented to connect the OSBridgeLCCA application and the report generator. The report bridge received input data from the application user interface through the command window and passed it to the report generator for PDF creation.

Implementation

The FinancialReportGenerator class reads the LaTeX template file and replaces placeholders such as logo path, analysis data, and calculated values. It then generates a .tex file and compiles it into a PDF using pdflatex.

The report bridge function generate_financial_pdf was implemented to act as an interface between the application and the report generator. It provides the template path, output directory, and logo path and calls the generator to create the final report.

The system successfully generated a financial Life Cycle Cost Analysis report in PDF format using input data from the application.

Outcome

The financial report generator and report bridge were successfully implemented. The system was able to automatically generate professional PDF reports using input data from the OSBridgeLCCA application.

3.3 Task 1: Python Code

3.3.1 Description of Financial Report Generator and Bridge

The financial report generator was developed using Python to automate the creation of Life Cycle Cost Analysis reports. The generator reads a LaTeX template file, replaces placeholders with input values, and compiles the file into a PDF report.

The report bridge function was developed to receive input data from the OSBridgeL-CCA application and pass it to the report generator.

3.3.2 Financial Report Generator Code

Listing 3.1: Financial Report Generator Code

```
%--------------------begin code-------------

# osbridgelcca/reporting/financial_report_generator.py
from pathlib import Path
import subprocess

class FinancialReportGenerator:
    def __init__(self, template_path: Path, output_dir: Path):
        self.template = template_path
        self.output_dir = output_dir
        self.output_dir.mkdir(parents=True, exist_ok=True)

    def generate(self, data, time_cost, logo_path, filename):
        tex_text = self.template.read_text()

        # Replace placeholders
        safe_logo_path = logo_path.replace("\\", "/")
        tex_text = tex_text.replace("<<LOGO_PATH>>", safe_logo_path)

        for key in data:
            tex_text = tex_text.replace(f"<<{key}>>", str(data[key]))
        tex_text = tex_text.replace("<<TIME_COST>>", str(time_cost))

        tex_file = self.output_dir / f"{filename}.tex"
```

```
25          tex_file.write_text(tex_text)
26
27          # Run pdflatex in output folder
28          subprocess.run(
29              ["pdflatex", "-interaction=nonstopmode", tex_file.name],
30              cwd=self.output_dir,
31              stdout=subprocess.DEVNULL,
32              stderr=subprocess.DEVNULL
33          )
34
35          pdf_path = self.output_dir / f"{filename}.pdf"
36          return str(pdf_path)
37
38  %-------------------- end code ---------------
```

3.3.3 Explanation of the Financial Report Generator Code

The FinancialReportGenerator class is responsible for generating the financial report PDF from the LaTeX template.

- **Line 1–2**: Imports the required modules Path for file handling and subprocess for executing the LaTeX compiler.
- **Line 4–8**: Defines the constructor method `__init__()` which initializes the template path and output directory. It also creates the output directory if it does not already exist.
- **Line 10**: Defines the generate() function, which is the main function used to generate the report.
- **Line 12**: Reads the LaTeX template file using the read_text() function.
- **Line 15–17**: Replaces the LOGO_PATH placeholder with the actual logo path.
- **Line 19–21**: Iterates through the input data dictionary and replaces each placeholder in the template with its corresponding value.
- **Line 22**: Replaces the TIME_COST placeholder with the calculated time cost value.

- **Line 24–25**: Creates and writes the modified LaTeX content to a new .tex file.
- **Line 28–33**: Executes the pdflatex command using subprocess to compile the .tex file and generate the PDF report.
- **Line 35–36**: Returns the path of the generated PDF file.

The generator automates the entire report generation process, converting template data into a professional PDF report.

3.3.4 Financial Report Bridge Code

Listing 3.2: Financial Report Bridge Code

```
1  %--------------------begin code-------------
2
3  # osbridgelcca/reporting/financial_report_bridge.py
4  from pathlib import Path
5  from .financial_report_generator import FinancialReportGenerator
6
7  def generate_financial_pdf(data, time_cost):
8      root = Path(__file__).resolve().parents[1]
9
10     template_path = root / "reports" / "templates" / "financial_report.
           tex"
11     output_dir = root / "reports" / "output"
12     logo_path = root / "desktop_app" / "resources" / "osbridge_logo.png
           "
13
14     generator = FinancialReportGenerator(template_path, output_dir)
15
16     return generator.generate(
17         data=data,
18         time_cost=time_cost,
19         logo_path=str(logo_path),
20         filename="financial_lcca_report"
21     )
22 %-------------------- end code ---------------
```

3.3.5 Explanation of the Report Bridge Code

The report bridge acts as a connection between the OSBridgeLCCA application and the financial report generator.

- **Line 1**: Imports the Path module for handling file paths.
- **Line 2**: Imports the FinancialReportGenerator class.
- **Line 4**: Defines the function generate_financial_pdf() which is used to generate the financial report.
- **Line 6**: Defines the root directory of the project.
- **Line 8**: Specifies the path of the LaTeX template file.
- **Line 9**: Defines the output directory where the generated report will be saved.
- **Line 10**: Defines the path to the logo file used in the report.
- **Line 12**: Creates an instance of the FinancialReportGenerator class.
- **Line 14–19**: Calls the generate() function and passes input data, time cost, logo path, and filename to generate the final PDF report.

The report bridge enables automatic report generation by connecting application input data with the report generator.

3.3.6 LaTeX Template Code

The LaTeX template was developed to generate a structured Life Cycle Cost Analysis financial report automatically. .

```
1 %--------------------begin code-------------
2
3 \usepackage[margin=1in]{geometry}
4 \usepackage{xcolor}
5 \usepackage{graphicx}
6 \usepackage{fancyhdr}
7 \usepackage{tabularx}
8
```

```
% --- Colors ---
\definecolor{osgreen}{HTML}{285A23}

% --- Header Style ---
\pagestyle{fancy}
\fancyhf{}
\renewcommand{\headrulewidth}{0pt}

% Logo at top-left
\fancyhead[L]{\includegraphics[height=1.4cm]{<<LOGO_PATH>>}}

% Footer (copyright + page no.)
\fancyfoot[C]{\textcolor{gray}{ OSBridge-LCCA     Page \thepage}}

\begin{document}

% space so title does NOT overlap logo
\vspace*{0.6cm}

% Title centered
\begin{center}
\textbf{\fontsize{18pt}{20pt}\selectfont LIFE CYCLE COST ANALYSIS
   REPORT}
\end{center}

\vspace{0.4cm}

% --- Banner ---
\begin{center}
\colorbox{osgreen}{%
    \parbox{0.8\linewidth}{%
        \centering
        \textcolor{white}{\textbf{FINANCIAL REPORT}}
    }
}
\end{center}

\vspace{0.4cm}

```

```
% ---- DATA TABLE CENTERED ----
\begin{center}
\begin{tabular}{|p{7cm}|p{5cm}|}
\hline
\textbf{Analysis Period} & <<Analysis Period>> years \\ \hline
\textbf{Design Life} & <<Design Life>> years \\ \hline
\textbf{Discount Rate (Inflation Adjusted)} & <<Discount Rate(Inflation
    Adjusted)>> \\ \hline
\textbf{Inflation Rate} & <<Inflation Rate>> \\ \hline
\textbf{Interest Rate} & <<Interest Rate>> \\ \hline
\textbf{Investment Ratio} & <<Investment Ratio>> \\ \hline
\textbf{Construction Duration} & <<Time for Construction of Base
   Project>> days \\ \hline
\end{tabular}
\end{center}

\vspace{0.3cm}

% --- Calculated Section ---
\section*{Calculated Values}
\textbf{Time-Cost:} <<TIME_COST>>

\vfill

{\centering
\textit{Generated automatically by OSBridge-LCCA} \par
}

%-------------------- end code ---------------
```

3.4 Task 1: Documentation

This section describes the financial report generation system developed for the OS-BridgeLCCA application.

3.4.1 Directory Structure

The directory structure of the financial report generation module is shown below:

```
osbridgelcca
├── reporting
│   ├── financial_report_bridge.py
│   └── financial_report_generator.py
├── reports
│   ├── templates
│   │   └── financial_report.tex
│   └── output
│       └── financial_lcca_report.pdf
└── desktop_app
    └── resources
        └── osbridge_logo.png
```

3.4.2 Component Description

LaTeX Template: Defines the report format with placeholders.

Report Generator: Reads the template, replaces placeholders, and generates the PDF.

Report Bridge: Transfers input data from the application to the report generator.

3.4.3 Output

The final report is generated in the output folder as:

`financial_lcca_report.pdf`

Chapter 4

Internship Task 2:Integration of LaTeX Report Template and Automated Report Generation in OSBridge-LCCA

4.1 Task 2: Problem Statement

The objective of this task was to integrate the developed LaTeX report template with the OSBridgeLCCA application and automate the report generation process. This involved implementing a report bridge and a financial report generator to retrieve input and calculated data from the application and generate a professional PDF report.

In addition, the existing report template was enhanced to improve its structure and presentation. Dynamic placeholders were added to include project details such as client name, job number, company name, and valuer name. The report header and footer were improved by adding the 3psLCCA format and IIT Bombay logo, with support for future customization.

The final goal was to ensure that the report is generated automatically when the user completes all steps in the application and clicks the Calculate button, and that the report is saved in the designated reports folder.

4.2 Task 2: Tasks Done

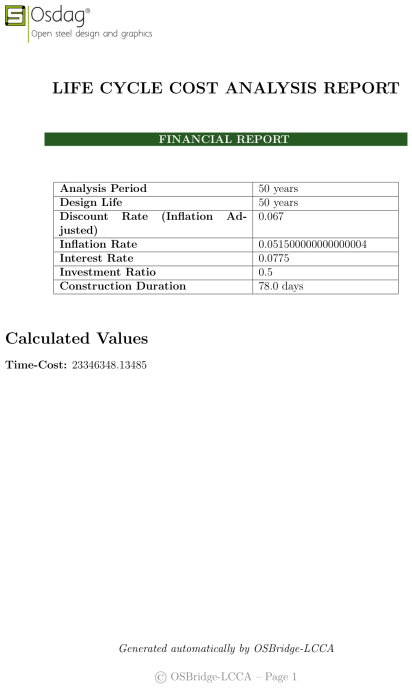
4.2.1 Enhancement of LaTeX Report Template

The existing Life Cycle Cost Analysi report template was modified to improve its layout, structure, and presentation. A new enhanced report format was developed under the 3psLCCA framework.

The following enhancements were implemented:

- Added dynamic 3psLCCA header and report title
- Integrated IIT Bombay logo with replaceable path for future modifications
- Added placeholders for project details such as Client Name, Job Number, Company Name, and Valuer Name
- Improved footer to display valuer name and company name on left and right sides
- Structured the report into proper sections including Financial Data and Cost Summary

Figure 4.1 shows the report before modification, and Figure 4.2 shows the enhanced report after modification.



Osdag
Open steel design and graphics

LIFE CYCLE COST ANALYSIS REPORT

FINANCIAL REPORT

Analysis Period	50 years
Design Life	50 years
Discount Rate (Inflation Adjusted)	0.067
Inflation Rate	0.051500000000000004
Interest Rate	0.0775
Investment Ratio	0.5
Construction Duration	78.0 days

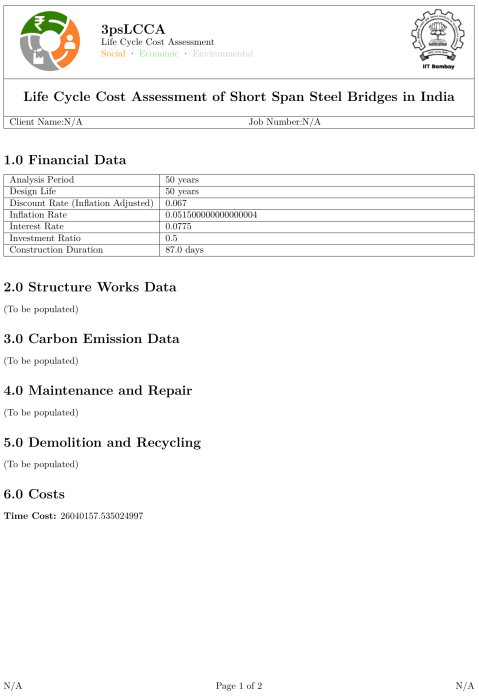
Calculated Values

Time-Cost: 23346348.13485

Generated automatically by OSBridge-LCCA

© OSBridge-LCCA – Page 1

Figure 4.1: Financial Report Before Modification



3psLCCA
Life Cycle Cost Assessment
Social · Economic · Environmental

IIT Bombay

Life Cycle Cost Assessment of Short Span Steel Bridges in India

Client Name:N/A Job Number:N/A

1.0 Financial Data

Analysis Period	50 years
Design Life	50 years
Discount Rate (Inflation Adjusted)	0.067
Inflation Rate	0.051500000000000004
Interest Rate	0.0775
Investment Ratio	0.5
Construction Duration	87.0 days

2.0 Structure Works Data

(To be populated)

3.0 Carbon Emission Data

(To be populated)

4.0 Maintenance and Repair

(To be populated)

5.0 Demolition and Recycling

(To be populated)

6.0 Costs

Time Cost: 26040157.535024997

N/A Page 1 of 2 N/A

Figure 4.2: Enhanced 3psLCCA Report After Modification

4.2.2 Updated LaTeX Template Code

The LaTeX template was enhanced to include dynamic logos, improved headers, structured sections, and automated data population for the Life Cycle Cost Analysis report.

Listing 4.1 shows the updated LaTeX template used in the 3psLCCA Report.

Listing 4.1: Updated Report Template

```
1
2  %--------------------begin code-------------
3
4  \usepackage[T1]{fontenc}
5  \usepackage[utf8]{inputenc}
6  \usepackage{lmodern}
7  \usepackage{geometry}
8  \usepackage{graphicx}
9  \usepackage{tabularx}
10 \usepackage[table]{xcolor}
11 \usepackage{fancyhdr}
12 \usepackage{array}
13 \usepackage{lastpage}
```

```
14
15 %---------------- Geometry ----------------
16 \geometry{
17   top=3cm,
18   bottom=2cm,
19   left=2cm,
20   right=2cm,
21   headheight=40pt,  % Increased to fit header
22   footskip=30pt,
23   includehead  % Include header in text area calculation
24 }
25
26 %---------------- Colors ----------------
27 \definecolor{EconomicPillar}{HTML}{89E88B}
28 \definecolor{EnvironmentalPillar}{HTML}{CCCCCC}
29 \definecolor{SocialPillar}{HTML}{FF9800}
30
31 %---------------- Header ----------------
32 \pagestyle{fancy}
33 \fancyhf{}
34 \renewcommand{\headrulewidth}{0pt}
35
36 \fancyfoot[C]{Page \thepage\ of \pageref{LastPage}}
37 \fancyfoot[L]{<<VALUER_NAME>>}
38 \fancyfoot[R]{<<COMPANY_NAME>>}
39
40 % ORIGINAL HEADER SIZE - FIXED table preamble
41 \fancyhead[L]{%
42 \raisebox{0pt}[0pt][0pt]{\begin{tabularx}{\textwidth}{|>{\centering\
      arraybackslash}m{3cm} >{\raggedright\arraybackslash}X >{\centering\
      arraybackslash}m{3cm}|}  % CHANGED: centering to raggedright
43 \hline
44 \includegraphics[height=2.4cm]{Logo3.png} &
45 \begin{tabular}{@{}l@{}}  % CHANGED: @{}c@{} to @{}l@{} for left
      alignment
46 \Large\textbf{3psLCCA}\\
47 Life Cycle Cost Assessment\\
48 {\color{SocialPillar}Social}    {\color{EconomicPillar}Economic}    {\
      color{EnvironmentalPillar}Environmental}
```

```
\end{tabular} &
\includegraphics[height=2.6cm]{iit_logo.png} \\ \hline

\multicolumn{3}{|c|}{%
\parbox[c][1.4cm][c]{\dimexpr\textwidth-2\tabcolsep-2\arrayrulewidth\relax}{%
\centering\Large\textbf{Life Cycle Cost Assessment of Short Span Steel Bridges in India}%
}}\\ \hline

\multicolumn{1}{|l}{Client Name:<<CLIENT_NAME>>} & \multicolumn{2}{c|}{Job Number:<<JOB_NUMBER>>} \\ \hline
\end{tabularx}}
}

%---------------- Document ----------------


\thispagestyle{fancy}

% Force the header to appear by starting text lower
\vspace*{0.6cm}

\section*{1.0 Financial Data}
\begin{tabularx}{\linewidth}{|l|X|}
\hline
Analysis Period & <<Analysis Period>> years \\ \hline
Design Life & <<Design Life>> years \\ \hline
Discount Rate (Inflation Adjusted) & <<Discount Rate(Inflation Adjusted)>> \\ \hline
Inflation Rate & <<Inflation Rate>> \\ \hline
Interest Rate & <<Interest Rate>> \\ \hline
Investment Ratio & <<Investment Ratio>> \\ \hline
Construction Duration & <<Time for Construction of Base Project>> days \\ \hline
\end{tabularx}

\vspace{0.4cm}

```

```
\section*{2.0 Structure Works Data}
(To be populated)

\section*{3.0 Carbon Emission Data}
(To be populated)

\section*{4.0 Maintenance and Repair}
(To be populated)

\section*{5.0 Demolition and Recycling}
(To be populated)

\section*{6.0 Costs}
\textbf{Time Cost:}    <<TIME_COST>>

\newpage
% MINIMAL FIX: Add space on second page for header
\vspace*{0.6cm}

\section*{7.0 Calculation Procedures}
(To be populated)

\section*{8.0 Summary}
(To be populated)

\end{document}

%----------------------end code---------------
```

4.2.3 Implementation of Report Generator

A Financial Report Generator was developed using Python to automate the report generation process. The generator reads the LaTeX template file, replaces placeholders with input and calculated values, and compiles the file into a PDF report using pdflatex.

This allows automatic generation of reports without manual editing.

4.2.4 Report Generator Code

The Financial Report Generator was implemented to automatically create the LaTeX file, replace placeholders with actual data, and compile it into a PDF report.

Listing 4.2 shows the report generator implementation.

Listing 4.2: Report Generator Code

```
1
2  %--------------------begin code-------------
3
4  from pathlib import Path
5  import subprocess
6
7  class FinancialReportGenerator:
8      def __init__(self, template_path: Path, output_dir: Path):
9          self.template_path = template_path
10         self.output_dir = output_dir
11         self.output_dir.mkdir(parents=True, exist_ok=True)
12
13     def generate(self, data, time_cost, logo_path, filename):
14         # 1. Read template
15         tex_text = self.template_path.read_text(encoding="utf-8")
16
17         # 2. Replace placeholders
18         replacements = {
19             "<<LOGO_PATH>>": logo_path.replace("\\", "/"),
20             "<<Analysis Period>>": str(data.get("Analysis Period", ""))
                   ,
21             "<<Design Life>>": str(data.get("Design Life", "")),
22             "<<Discount Rate(Inflation Adjusted)>>": str(data.get("
                   Discount Rate(Inflation Adjusted)", "")),
23             "<<Inflation Rate>>": str(data.get("Inflation Rate", "")),
24             "<<Interest Rate>>": str(data.get("Interest Rate", "")),
25             "<<Investment Ratio>>": str(data.get("Investment Ratio", ""
                   )),
26             "<<Time for Construction of Base Project>>": str(data.get("
                   Time for Construction of Base Project", "")),
27             "<<TIME_COST>>": f"{time_cost:,.2f}",
28         }
```

```
29
30          for key, value in replacements.items():
31              tex_text = tex_text.replace(key, value)
32
33          # 3. Write FILLED tex file
34          tex_file = self.output_dir / f"{filename}.tex"
35          tex_file.write_text(tex_text, encoding="utf-8")
36
37          # 4. Compile PDF
38          subprocess.run(
39              ["pdflatex", "-interaction=nonstopmode", tex_file.name],
40              cwd=self.output_dir,
41              check=True
42          )
43
44          return self.output_dir / f"{filename}.pdf"
45
46
47  %--------------------end code--------------
```

4.2.5 Implementation of Report Bridge

A Report Bridge was implemented to connect the OSBridgeLCCA application and the report generator.

The report bridge retrieves the following data from the application database:

- Project Details
- Financial Data
- Calculation Results

This data is passed to the report generator to create the final PDF report.

4.2.6 Report Bridge Code

The report bridge connects the OSBridge-LCCA application interface with the report generator and passes the required input parameters.

Listing 4.X shows the report bridge implementation.

Listing 4.3: Financial Report Bridge Code

```
1
2  %--------------------begin code-------------
3
4  from pathlib import Path
5  import subprocess
6
7  def generate_full_pdf(database_manager):
8
9      # ---- Pull stored data ----
10     project = database_manager.project_details
11     financial = database_manager.financial_data
12     results = database_manager.results
13
14     data = {
15         # ---- Project details ----
16         "CLIENT_NAME": project.get("CLIENT_NAME", "N/A"),
17         "JOB_NUMBER": project.get("JOB_NUMBER", "N/A"),
18         "COMPANY_NAME": project.get("COMPANY_NAME", "N/A"),
19         "VALUER_NAME": project.get("VALUER_NAME", "N/A"),
20
21         # ---- Financial data ----
22         "Analysis Period": financial.get("Analysis Period", "N/A"),
23         "Design Life": financial.get("Design Life", "N/A"),
24         "Discount Rate(Inflation Adjusted)": financial.get("Discount
               Rate(Inflation Adjusted)", "N/A"),
25         "Inflation Rate": financial.get("Inflation Rate", "N/A"),
26         "Interest Rate": financial.get("Interest Rate", "N/A"),
27         "Investment Ratio": financial.get("Investment Ratio", "N/A"),
28         "Time for Construction of Base Project": financial.get("Time
               for Construction of Base Project", "N/A"),
29
30         # ---- Cost ----
31         "TIME_COST": results.get("Time Cost", "N/A"),
32     }
33
34     # ---- Paths ----
35     root = Path(__file__).resolve().parents[1]
```

```
    template_path = root / "reports" / "templates" / "financial_report.
        tex"
    output_dir = root / "reports" / "output"
    output_dir.mkdir(exist_ok=True)

    tex_file = output_dir / "financial_report.tex"

    # ---- Replace placeholders ----
    tex = template_path.read_text(encoding="utf-8")

    for key, value in data.items():
        tex = tex.replace(f"<<{key}>>", str(value))

    tex_file.write_text(tex, encoding="utf-8")

    # ---- Compile PDF ----
    subprocess.run(
        ["pdflatex", "-interaction=nonstopmode", tex_file.name],
        cwd=output_dir
    )

    print("       PDF generated successfully")

%--------------------end code--------------
```

4.2.7 Integration with OSBridgeLCCA Application

The report generation system was fully integrated into the OSBridgeLCCA workflow.

The application consists of multiple steps where the user enters required project and financial data.

After completing all steps, when the user clicks the **Calculate** button in the final step, the Life Cycle Cost Analysis calculations are performed and the report generation process is automatically triggered.

The report bridge retrieves the data from the database and passes it to the report generator.

The report generator fills the LaTeX template and compiles the PDF report.

The final report is automatically saved in the designated reports folder inside the src directory of the OSBridgeLCCA project.

This ensures fully automated report generation as part of the application workflow.

4.3 Task 2: Documentation

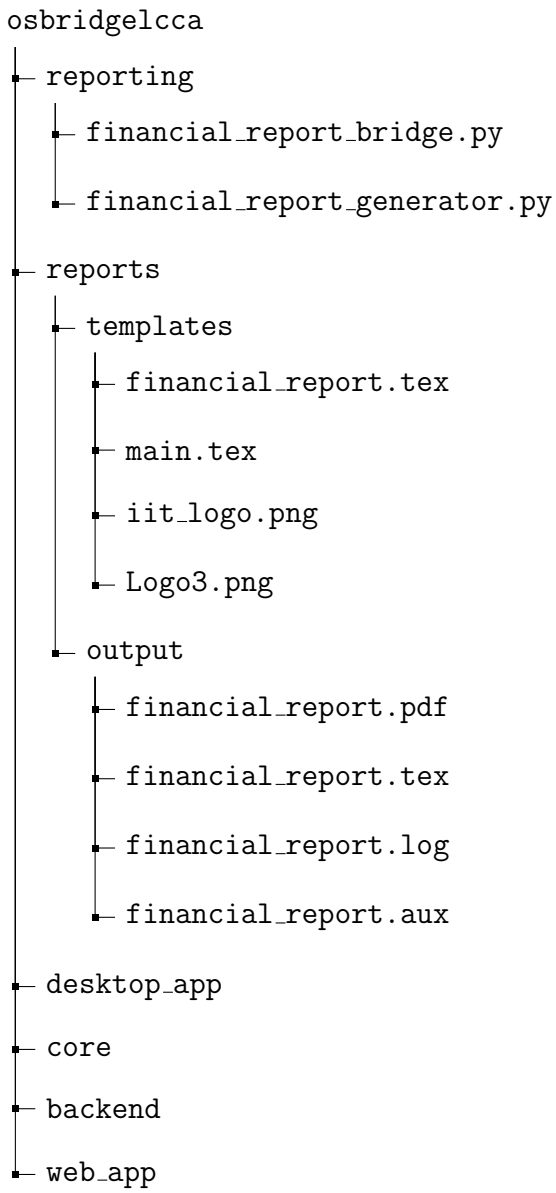
4.3.1 Documentation

This section provides the documentation of the automated report generation system integrated with the OSBridge-LCCA application.

- The LaTeX template was developed to provide a structured financial report format.
- The template uses placeholders to insert input parameters and calculated values.
- The report generator module processes the template and replaces placeholders with actual data.
- The report bridge module transfers data from the OSBridge-LCCA application to the generator.
- The system automatically compiles the LaTeX file to generate the final PDF report.
- The report includes customized headers, footers, and logos for proper presentation.
- The generated report is automatically saved in the reports folder of the application.
- The system ensures consistent, accurate, and automated report generation.

4.3.2 Directory Structure

The directory structure of the report generation module in the OSBridgeLCCA project is shown below:



4.3.3 Report Generation Workflow

The automated report generation process follows these steps:

- The user enters project and financial data using the OSBridgeLCCA graphical interface.
- After completing all required inputs, the user clicks the Calculate button.
- The application performs Life Cycle Cost Analysis calculations.
- The Financial Report Bridge module retrieves the input data and calculated results.

- The Financial Report Generator replaces placeholders in the financial_report.tex template.
- The LaTeX template is compiled automatically to generate the final PDF report.
- The generated report is saved in the reports/output directory.

4.3.4 Template Integration

The report template was enhanced with the following features:

- Dynamic insertion of financial and project data
- Integration of OSBridgeLCCA and IIT logos
- Improved header and footer formatting
- Professional report layout and structure
- Automatic generation of calculated values section

4.3.5 Output Files

After report generation, the following files are created in the output directory:

- financial_report.pdf – Final generated report
- financial_report.tex – Processed LaTeX file
- financial_report.log – Compilation log file
- financial_report.aux – LaTeX auxiliary file

This implementation ensures fully automated report generation integrated with the OSBridgeLCCA workflow.

Chapter 5

Conclusions

5.1 Tasks Accomplished

During the course of this internship, significant contributions were made towards understanding, testing, and improving the reporting functionality of the Osdag and OSBridgeLCCA applications. The major tasks accomplished are summarized below:

- Performed unit testing of various modules in the Osdag application, including steel connection design modules. Multiple test cases were executed to verify correctness and stability.
- Studied the OSBridgeLCCA application workflow and analyzed the existing Life Cycle Cost Analysis report generation system.
- Designed and developed a structured LaTeX report template with improved formatting, headers, logos, and organized financial data presentation.
- Implemented a Financial Report Generator using Python to automatically generate PDF reports from input data.
- Developed a Report Bridge module to integrate the OSBridgeLCCA application with the LaTeX report generator.
- Successfully automated the report generation process, where the final report is generated and saved automatically in the reports folder when the user performs calculations in the application.

These tasks enhanced the overall report generation system and improved the usability and automation capability of the application.

5.2 Skills Developed

This internship provided valuable technical and professional learning experience. The key skills developed are listed below:

Technical Skills:

- Gained practical experience in software testing and validation.
- Learned LaTeX for professional report design and generation.
- Developed Python programming skills for automation and integration.
- Understood project structure and software architecture of large applications.
- Learned automated report generation using LaTeX and Python.

Professional Skills:

- Improved problem-solving and debugging ability.
- Developed technical documentation skills.
- Understood real-world software development workflow.
- Improved analytical and logical thinking skills.

Overall, the internship provided valuable exposure to real-world software development, testing, and report automation.

Chapter A

Appendix

A.1 Work Reports

Internship Work Report

Name:		**Ishan Rai**	
Project:		**OSDAG**	
Internship:		**FOSSEE Winter Internship 2025**	
DATE	**DAY**	**TASK**	**Hours Worked**
01-Dec-2025	Monday	Attended introductory meeting, cloned Osdag repository and explored structure	5
02-Dec-2025	Tuesday	Installed Osdag and studied documentation	5
03-Dec-2025	Wednesday	Explored Osdag source code and connection modules	5
04-Dec-2025	Thursday	Studied report generation workflow and LaTeX template	4
05-Dec-2025	Friday	Studied Header Plate and End Plate connection module	5
06-Dec-2025	Saturday	Attended Osdag Mass Testing meeting and performed testing	6
07-Dec-2025	Sunday	Continued mass testing and verified reports	5
08-Dec-2025	Monday	Documented testing results and observations	4
09-Dec-2025	Tuesday	Studied Osdag report generation system	5
10-Dec-2025	Wednesday	Completed Osdag repository understanding	4
11-Dec-2025	Thursday	Attended project meeting and assigned OSBridge-LCCA project	6
12-Dec-2025	Friday	Installed OSBridge-LCCA and explored modules	5
13-Dec-2025	Saturday	Attended LCC Meeting-2 and studied project workflow	4
14-Dec-2025	Sunday	Studied financial report template structure	5
15-Dec-2025	Monday	Started developing LaTeX report template	6
16-Dec-2025	Tuesday	Attended LCC Meeting-4 and improved template	4
17-Dec-2025	Wednesday	Added placeholders and tested template	5
18-Dec-2025	Thursday	Attended LCC Meeting-5 and improved formatting	4
19-Dec-2025	Friday	Tested template and fixed bugs	5
20-Dec-2025	Saturday	Studied Python report generation automation	4
21-Dec-2025	Sunday	Started developing report generator code	6
22-Dec-2025	Monday	Implemented automatic report generation logic	5
23-Dec-2025	Tuesday	Attended LCC Meeting-6 and debugged errors	4
24-Dec-2025	Wednesday	Improved report generation automation	5
25-Dec-2025	Thursday	Tested report generation functionality	4
26-Dec-2025	Friday	Attended LCC Meeting-7 and fixed generator bugs	4
27-Dec-2025	Saturday	Improved report layout and design	6
28-Dec-2025	Sunday	Developed report bridge code	5
29-Dec-2025	Monday	Attended LCC Meeting-8 and integrated system	4
30-Dec-2025	Tuesday	Debugged integration issues	5
31-Dec-2025	Wednesday	Attended LCC Meeting-9 and tested system	4
01-Jan-2026	Thursday	Improved report header and logo	4
02-Jan-2026	Friday	Added dynamic fields and formatting	5
03-Jan-2026	Saturday	Improved template structure	4
04-Jan-2026	Sunday	Tested report generation system	5
05-Jan-2026	Monday	Fixed template errors	5
06-Jan-2026	Tuesday	Attended LCC Meeting-10 and improved integration	4
07-Jan-2026	Wednesday	Debugged report generation bugs	6
08-Jan-2026	Thursday	Attended LCC Meeting-11 and tested system	4
09-Jan-2026	Friday	Fixed report generator bugs	5
10-Jan-2026	Saturday	Improved automation system	4
11-Jan-2026	Sunday	Performed testing and validation	5
12-Jan-2026	Monday	Attended LCC Meeting-12 and verified reports	4
13-Jan-2026	Tuesday	Fixed bugs and improved generator	5

14-Jan-2026	Wednesday	Attended project meeting and implemented feedback	4
15-Jan-2026	Thursday	Debugged system errors	5
16-Jan-2026	Friday	Attended LCC Meeting-14 and tested improvements	4
17-Jan-2026	Saturday	Performed system testing	5
18-Jan-2026	Sunday	Verified generated reports	4
19-Jan-2026	Monday	Attended LCC Meeting-15 and discussed progress	4
20-Jan-2026	Tuesday	Exam Break	0
21-Jan-2026	Wednesday	Exam Break	0
22-Jan-2026	Thursday	Exam Break	0
23-Jan-2026	Friday	Exam Break	0
24-Jan-2026	Saturday	Exam Break	0
25-Jan-2026	Sunday	Exam Break	0
26-Jan-2026	Monday	Fixed report integration bugs	5
27-Jan-2026	Tuesday	Attended LCC Meeting-19 and implemented feedback	4
28-Jan-2026	Wednesday	Debugged integration errors	6
29-Jan-2026	Thursday	Tested report system	5
30-Jan-2026	Friday	Fixed bugs and tested system	4
31-Jan-2026	Saturday	Verified generated reports	4
01-Feb-2026	Sunday	Performed final system testing	5
02-Feb-2026	Monday	Fixed minor bugs	4
03-Feb-2026	Tuesday	Validated report generation	5
04-Feb-2026	Wednesday	Verified integration	4
05-Feb-2026	Thursday	Fixed final bugs	4
06-Feb-2026	Friday	Started writing internship report	6
07-Feb-2026	Saturday	Wrote report implementation sections	5
08-Feb-2026	Sunday	Added code and explanations	4
09-Feb-2026	Monday	Added figures and diagrams	4
10-Feb-2026	Tuesday	Completed report writing	4
11-Feb-2026	Wednesday	Improved formatting	4
12-Feb-2026	Thursday	Added bibliography and conclusion	4
13-Feb-2026	Friday	Reviewed report	4
14-Feb-2026	Saturday	Corrected report errors	4
15-Feb-2026	Sunday	Prepared final report	4
16-Feb-2026	Monday	Final review	4

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.