FOSSEE Semester-Long Internship Report

On

Development of UI for OsBLCCA Tool for Osdag

Submitted by

Om Lakshkar

3rd Year B.Tech Student, Department of Computer Science and Engineering

GLA University

Mathura

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

June 7, 2025

Acknowledgments

I would like to begin by expressing my heartfelt gratitude to everyone who supported and guided me throughout my internship journey. This project has been an immensely rewarding experience, and I am truly thankful to all those who contributed to its success.

I am deeply grateful to the **FOSSEE team at IIT Bombay** for allowing me to work on a project that aligns with my interests and career aspirations. The internship selection through a screening task-based process was both challenging and enriching, and I am honored to have been a part of this initiative.

I would like to sincerely thank the dedicated project staff at the **Osdag team**, including **Mr. Ajmal Babu M. S., Mr. Ajinkya Dahale, and Mr. Parth Karia**, for their valuable guidance, constant support, and thoughtful mentorship throughout the internship period. Their commitment to fostering a collaborative and welcoming environment made this experience truly enjoyable and productive.

I extend my deepest gratitude to **Prof. Siddhartha Ghosh**, **Principal Investigator (PI)** of the Osdag project, Department of Civil Engineering, IIT Bombay, for his visionary leadership and support. I would also like to thank **Prof. Kannan M. Moudgalya**, **FOSSEE Project Investigator**, Department of Chemical Engineering, IIT Bombay, whose efforts have been instrumental in promoting the use of open-source software in education and research.

A special thanks to **Ms. Usha Viswanathan** and **Ms. Vineeta Parmar**, the FOSSEE Project Managers, and their entire team for managing the internship program with such efficiency and dedication.

I gratefully acknowledge the support from the **National Mission on Education through Information and Communication Technology (ICT)**, **Ministry of Education (MoE), Government of India**, whose funding and vision made this project possible.

This project was a collaborative effort, and I am thankful to have worked alongside my colleague **Souhridya Patra**. Together, we developed the user interface for the **OsBLCCA tool**, contributing to the open-source ecosystem that Osdag promotes.

This internship has been a remarkable learning experience, and I am truly honored to have contributed to a project that advances open-source software for educational and industrial use in our country.

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 Li-brary	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.

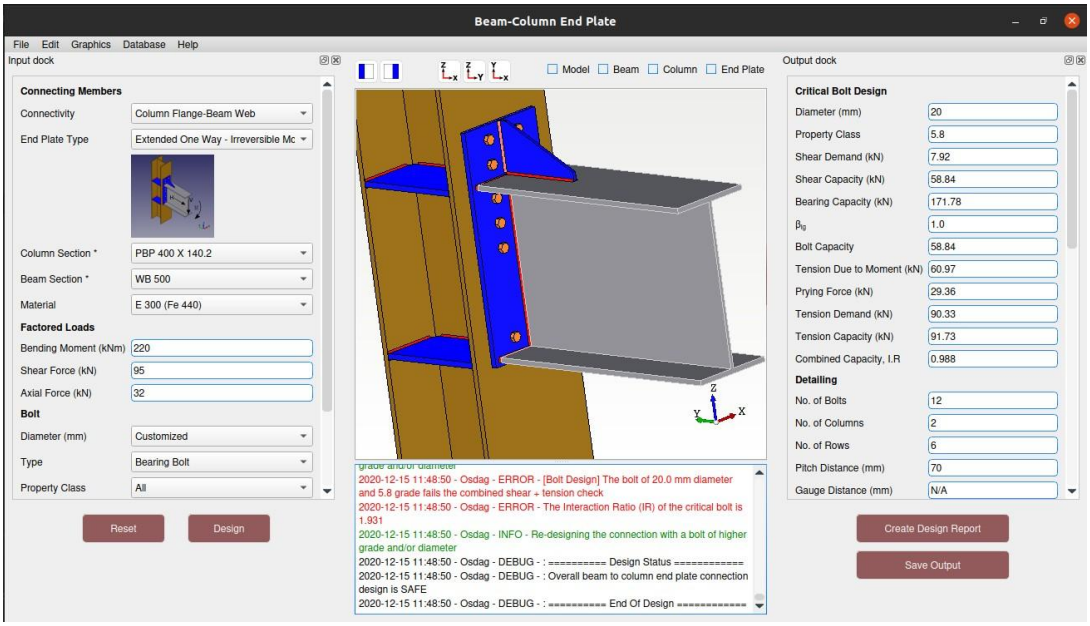


Figure 1.2: Osdag GUI

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

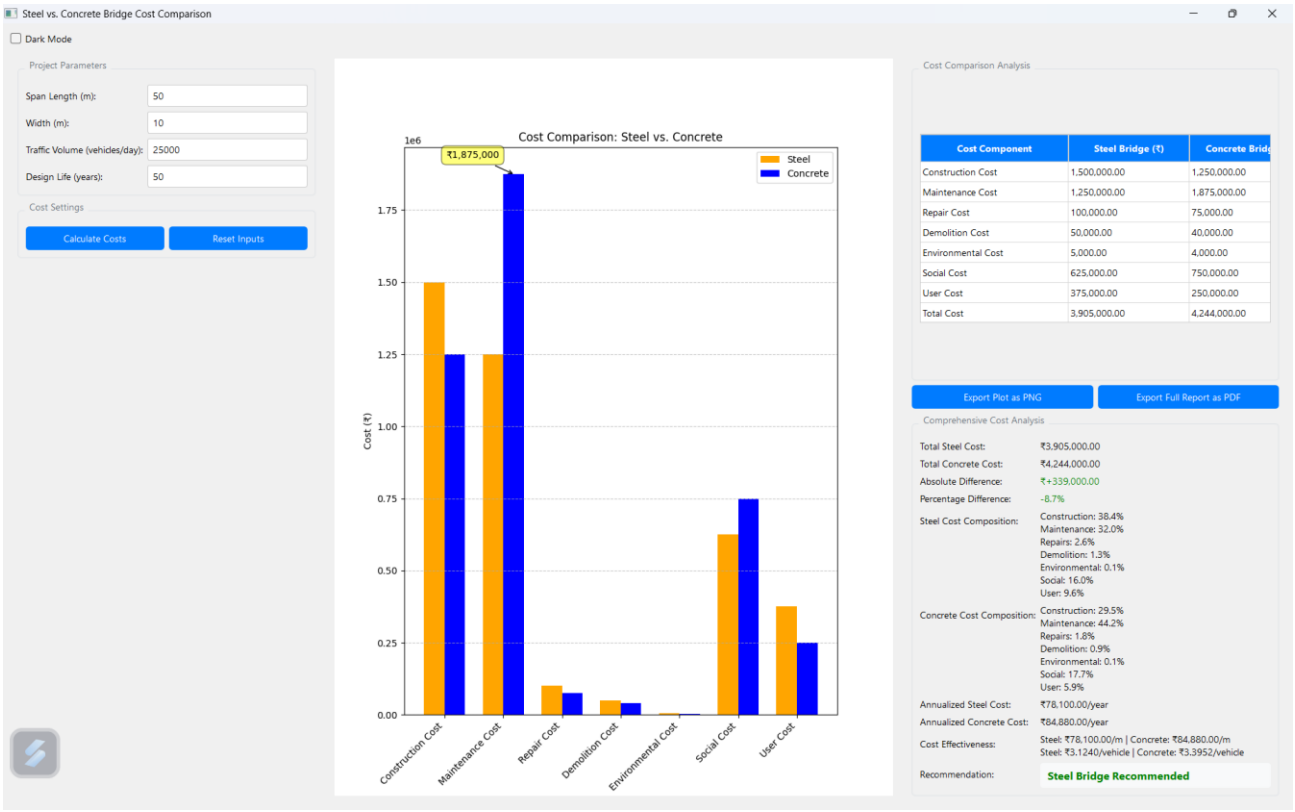
Screening Task

2.1 Problem Statement

Beam to Beam End Plate

2.2 Tasks Done

Developed a Desktop app for the same.



Chapter 3

Internship Task 1 Title

3.1Task 1: Problem Statement

The OsBLCCA tool required a functional and user-friendly GUI to make its features accessible to users. While design wireframes were available, they lacked implementation. The task was to transform these static designs into a responsive and professional desktop application interface using Python-based GUI frameworks.

3.2 Task 1: Tasks Done

You can look at all our work in this Git repo:- https://github.com/Omlakshkar07/Osdag_UI

1. **Tools and Technologies** Used To accomplish the given tasks, the following tools and technologies were utilized:
   • Python
   • PyQt5 / PySide2 – For GUI design and application structure
   • Qt Designer – To expedite layout creation
   • Git & GitHub – For version control and collaboration
   • VS Code / PyCharm – As the primary development environment

2. **Understanding the Wireframes**
   The document titled "Wireframes for OsBLCCA" served as the blueprint for UI development. It outlined various components of the software's graphical interface, including:
   • Welcome and Navigation Screen
   • Input Parameter Forms (Bridge Type, Material Type, Initial Cost, Maintenance Cost, etc.)
   • Results Display Window (Comparison charts, tables)
   • Export Options (Save as PDF/Excel)
   • Help and About Sections
   The wireframes were detailed enough to offer guidance on layout, field placements, and navigation flow, which significantly helped in the UI construction phase.

4.Implementation Strategy

The implementation of the UI was planned and executed in structured phases to ensure clarity, efficiency, and adherence to the design document. Each stage contributed incrementally to the final product.

Phase 1: Requirement Understanding and Tool Setup

• Analyzed the wireframe PDF document thoroughly to understand the expected layout, flow, and element placement.
• Installed and configured essential tools: Python, PyQt5, Qt Designer, and Git.
• Set up a local development environment using Visual Studio Code/PyCharm.

Phase 2: Layout Design and Structure Creation

• Recreated the structural layout of each screen from the wireframe using QMainWindow, QDialog, QTabWidget, and QVBoxLayout/QHBoxLayout classes.
• Ensured that spacing, alignment, and grouping of components matched the blueprint exactly.
• Added branding placeholders such as the project logo and headers.

Phase 3: Widget Integration

Integrated various widgets such as:

- QComboBox for bridge and material selection.
- QLineEdit for numeric and text input.
- QRadioButton and QCheckBox for conditional selections.
- QTableWidget for data display and summary.
- QPushButton for navigation, form submission, and export options.

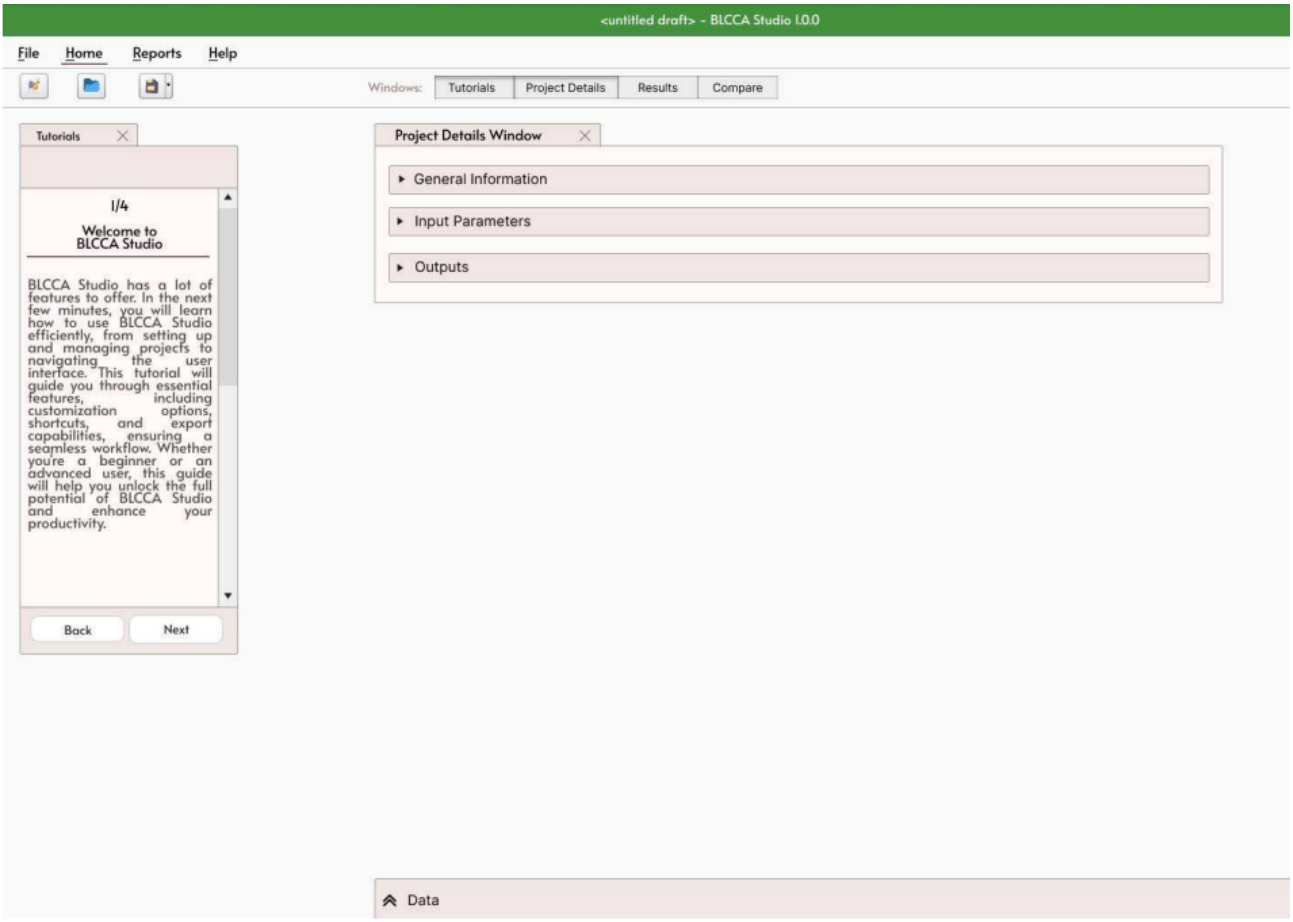
Created clear labels for each form field as per the wireframes and grouped them logically using QGroupBox.

Phase 4: Navigation and Flow Control

• Implemented signal-slot mechanisms to connect buttons with their respective functions like switching tabs, clearing forms, or moving between screens.
• Added placeholder functions for export (PDF/Excel), calculation, and help sections for future backend integration.

Phase 5: Responsive Testing and Adjustments

• Tested the GUI for different screen resolutions and font sizes.
• Ensured widget resizing and layout stretching to prevent misalignment or overflow.
• Used stylesheet (QSS) customization to improve UI appearance and maintain visual consistency.


<untitled draft> - BLCCA Studio 1.0.0
File
Home
Reports
Help
Windows:
Tutorials
Project Details
Results
Compare
Tutorials
1/4
Welcome to BLCCA Studio
BLCCA Studio has a lot of features to offer. In the next few minutes, you will learn how to use BLCCA Studio efficiently, from setting up and managing projects to navigating the user interface. This tutorial will guide you through essential features, including customization options, shortcuts, and export capabilities, ensuring a seamless workflow. Whether you're a beginner or an advanced user, this guide will help you unlock the full potential of BLCCA Studio and enhance your productivity.
Back
Next
Project Details Window
General Information
Input Parameters
Outputs
Data

3.3 Task 1: Python Code

This Python script builds a desktop GUI application using PyQt5 called BICCA Studio, which provides a user-friendly interface for managing project details, navigating tutorial pages, and interacting with toolbar/menu components. It includes menus, tabs, collapsible input sections, and tutorial navigation to guide users through the software's features.of the software.

3.3.1 Python Code

```
import sys
from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QWidget, QVBoxLayout, QHBoxLayout,
    QLabel, QPushButton, QTabWidget, QTreeWidget, QTreeWidgetItem,
    QSplitter, QFrame, QSizePolicy, QAction, QToolBar, QStackedWidget,
    QDockWidget, QScrollArea, QMenu, QLineEdit, QGroupBox
)
from PyQt5.QtCore import Qt, QSize, QPoint
from PyQt5.QtGui import QIcon, QFont, QColor, QPalette, QPixmap


class CloseableTabWidget(QTabWidget):
    """Custom tab widget with closeable tabs"""

    def __init__(self, parent=None):
        super(CloseableTabWidget, self).__init__(parent)
        self.setTabsClosable(True)
        self.tabCloseRequested.connect(self.closeTab)
        self.setDocumentMode(True)
        self.setMovable(True)

    def closeTab(self, index):
        """Close the tab at the given index"""
```

```
        # Don't actually remove tabs in this demo
        print(f"Close tab requested for tab index {index}")

class TreePanel(QWidget):
    """Project details tree panel implementation"""

    def __init__(self, parent=None):
        super(TreePanel, self).__init__(parent)
        self.initUI()

    def initUI(self):
        layout = QVBoxLayout(self)
        layout.setContentsMargins(0, 0, 0, 0)

        # Create header
        header = QFrame()
        header.setFrameShape(QFrame.StyledPanel)
        # Updated header background color to match the image (light brown/beige)
        header.setStyleSheet("background-color: #F5EFE6;")
        header_layout = QHBoxLayout(header)
        header_layout.setContentsMargins(5, 5, 5, 5)

        title_label = QLabel("Project Details Window")
        title_label.setStyleSheet("color: #333333; font-weight: bold;")
        header_layout.addWidget(title_label)

        close_btn = QPushButton("×")
        close_btn.setFixedSize(20, 20)
```

```
close_btn.setStyleSheet("""
    QPushButton {
        border: none;
        background-color: transparent;
        color: #555555;
        font-weight: bold;
        font-size: 14px;
    }
    QPushButton:hover {
        color: #000000;
    }
""")
header_layout.addWidget(close_btn)

layout.addWidget(header)

# Create tree widget
self.tree = QTreeWidget()
self.tree.setHeaderHidden(True)
self.tree.setStyleSheet("""
    QTreeWidget {
        border: none;
        background-color: #F5EFE0;  /* Beige background to match image */
    }
    QTreeWidget::item {
        height: 24px;
    }
    QTreeWidget::item:selected {
```

```
            background-color: #E6F2E6;
            color: #000000;
        }
        QTreeWidget::item:has-children {
            font-weight: bold;
            color: #333333;
        }
    """)

    # Input Parameters section
    input_params = QTreeWidgetItem(["Input Parameters"])
    input_params.setIcon(0, QIcon.fromTheme("folder", self.getDefaultIcon()))
    input_params.setBackground(0, QColor("#F5EFE0"))  # Set background for Input Parameters
    input_params.setForeground(0, QColor("#333333"))  # Dark text color
    self.tree.addTopLevelItem(input_params)

    # Structure Works Data
    structure_works = QTreeWidgetItem(["Structure Works Data"])
    structure_works.setIcon(0, QIcon.fromTheme("folder", self.getDefaultIcon()))
    structure_works.setForeground(0, QColor("#333333"))  # Dark text color
    input_params.addChild(structure_works)

    structure_works.addChild(self.createItem("Financial Data"))
    structure_works.addChild(self.createItem("Carbon Emission Data"))
    structure_works.addChild(self.createItem("Bridge and Traffic Data"))
    structure_works.addChild(self.createItem("Maintenance and Repair"))
    structure_works.addChild(self.createItem("Demolition and Recycling"))
```

```
# Direct children of Input Parameters
input_params.addChild(self.createItem("Carbon Emission Data"))
input_params.addChild(self.createItem("Bridge and Traffic Data"))
input_params.addChild(self.createItem("Maintenance and Repair"))
input_params.addChild(self.createItem("Demolition and Recycling"))

# Output section
output = QTreeWidgetItem(["Output"])
output.setIcon(0, QIcon.fromTheme("folder", self.getDefaultIcon()))
output.setBackground(0, QColor("#F5EFE0"))  # Set background for Output
output.setForeground(0, QColor("#333333"))  # Dark text color
self.tree.addTopLevelItem(output)

# Output items
output_items = [
    "Initial Construction Cost",
    "Initial Carbon Emission Cost",
    "Time Cost",
    "Traffic User Cost",
    "Carbon Emission due to Re-routing",
    "Periodic Maintenance Costs",
    "Maintenance Emission Costs",
    "Routine Inspection Costs",
    "Repair & Rehabilitation Costs",
    "Reconstruction Costs",
    "Demolition & Disposal Cost",
    "Recycling Cost",
```

```
        "Total Life-Cycle Cost"
    ]

    for item_text in output_items:
        item = self.createItem(item_text, is_document=True)
        output.addChild(item)

    # Expand input params by default
    input_params.setExpanded(True)
    structure_works.setExpanded(True)
    output.setExpanded(True)

    layout.addWidget(self.tree)

def createItem(self, text, is_document=False):
    """Create a tree item with the appropriate icon"""
    item = QTreeWidgetItem([text])
    if is_document:
        # Document icon for output items
        item.setIcon(0, QIcon.fromTheme("text-x-generic", self.getDocumentIcon()))
    else:
        # Default folder icon
        item.setIcon(0, QIcon.fromTheme("folder", self.getDefaultIcon()))
    return item

def getDefaultIcon(self):
    """Return a default folder icon"""
    pixmap = QPixmap(16, 16)
```

```
        pixmap.fill(QColor(0, 0, 0, 0))
        return QIcon(pixmap)

    def getDocumentIcon(self):
        """Return a default document icon"""
        pixmap = QPixmap(16, 16)
        pixmap.fill(QColor(0, 0, 0, 0))
        return QIcon(pixmap)

class DataWindowPanel(QWidget):
    """Data Window Panel Implementation"""

    def __init__(self, parent=None):
        super(DataWindowPanel, self).__init__(parent)
        self.initUI()

    def initUI(self):
        layout = QVBoxLayout(self)

        # Placeholder for chart area - upper section
        chart_placeholder = QWidget()
        chart_placeholder.setMinimumHeight(300)
        chart_placeholder.setStyleSheet("background-color: #FFFFFF; border: 1px solid
#CCCCCC;")

        placeholder_text = QLabel("Life-Cycle Costs for 50 years\n\nVisualization area for donut
charts")
        placeholder_text.setAlignment(Qt.AlignCenter)
```

```
chart_layout = QVBoxLayout(chart_placeholder)
chart_layout.addWidget(placeholder_text)

layout.addWidget(chart_placeholder)

# Create legend
legend_frame = QFrame()
legend_frame.setFrameShape(QFrame.StyledPanel)
legend_frame.setMaximumHeight(50)
legend_frame.setStyleSheet("background-color: #F5F5F5; border: 1px solid #DDDDDD;")

legend_layout = QHBoxLayout(legend_frame)

legend_items = [
    ("Initial Stage", "#000000"),
    ("Use Stage", "#006400"),
    ("End-of-Life Stage", "#8B4513"),
    ("Beyond-Life Stage", "#90EE90")
]

for text, color in legend_items:
    color_box = QFrame()
    color_box.setFixedSize(16, 16)
    color_box.setStyleSheet(f"background-color: {color}; border: 1px solid black;")

    label = QLabel(text)

    item_layout = QHBoxLayout()
```

```
            item_layout.setContentsMargins(0, 0, 0, 0)
            item_layout.addWidget(color_box)
            item_layout.addWidget(label)

            legend_layout.addLayout(item_layout)

        legend_layout.addStretch()
        layout.addWidget(legend_frame)

        # Placeholder for horizontal bar chart - lower section
        barchart_placeholder = QWidget()
        barchart_placeholder.setMinimumHeight(300)
        barchart_placeholder.setStyleSheet("background-color: #FFFFFF; border: 1px solid
#CCCCCC;")

        barchart_text = QLabel("Horizontal bar chart area showing breakdown of life-cycle costs")
        barchart_text.setAlignment(Qt.AlignCenter)
        barchart_layout = QVBoxLayout(barchart_placeholder)
        barchart_layout.addWidget(barchart_text)

        layout.addWidget(barchart_placeholder)

        # Total cost display
        total_frame = QFrame()
        total_layout = QHBoxLayout(total_frame)
        total_layout.setContentsMargins(5, 5, 5, 5)

        total_layout.addStretch()
```

```
total_label = QLabel("Total Life-Cycle Cost:")
total_label.setStyleSheet("font-weight: bold; font-size: 12px;")
total_layout.addWidget(total_label)

total_value = QLabel("259.15 Lakh")
total_value.setStyleSheet("font-weight: bold; font-size: 12px;")
total_layout.addWidget(total_value)

layout.addWidget(total_frame)

# Download and view options
options_frame = QFrame()
options_layout = QVBoxLayout(options_frame)

download_layout = QHBoxLayout()
download_layout.addWidget(QPushButton("Download as PNG"))
download_layout.addWidget(QPushButton("Download as JPG"))
download_layout.addWidget(QPushButton("Download as PDF"))
options_layout.addLayout(download_layout)

view_layout = QHBoxLayout()
view_layout.addWidget(QPushButton("View as Pie Chart"))
view_layout.addWidget(QPushButton("View as Table"))
options_layout.addLayout(view_layout)

layout.addWidget(options_frame)
```

```
        # Navigation buttons
        nav_frame = QFrame()
        nav_layout = QHBoxLayout(nav_frame)

        nav_layout.addStretch()

        back_btn = QPushButton("Back")
        back_btn.setFixedWidth(100)
        nav_layout.addWidget(back_btn)

        next_btn = QPushButton("Next")
        next_btn.setFixedWidth(100)
        nav_layout.addWidget(next_btn)

        layout.addWidget(nav_frame)

class ResultsWindowPanel(QWidget):
    """Results Window Panel Implementation"""

    def __init__(self, parent=None):
        super(ResultsWindowPanel, self).__init__(parent)
        self.initUI()

    def initUI(self):
        layout = QVBoxLayout(self)
        layout.setContentsMargins(0, 0, 0, 0)

        # Create tabbed widget for Economic, Social, Environmental costs
```

```
tabs = QTabWidget()

tabs.setDocumentMode(True)


# Economic cost tab

economic_tab = QWidget()

economic_layout = QVBoxLayout(economic_tab)


cost_title = QLabel("Economic cost distribution across various stages for bridges for 50 years")

cost_title.setAlignment(Qt.AlignCenter)

economic_layout.addWidget(cost_title)


chart_placeholder = QWidget()

chart_placeholder.setMinimumHeight(300)

chart_placeholder.setStyleSheet("background-color: #FFFFFF; border: 1px solid #CCCCCC;")


chart_text = QLabel("Economic cost distribution donut chart")

chart_text.setAlignment(Qt.AlignCenter)

chart_layout = QVBoxLayout(chart_placeholder)

chart_layout.addWidget(chart_text)


economic_layout.addWidget(chart_placeholder)


# Download options

dl_layout = QHBoxLayout()

dl_layout.addWidget(QPushButton("Download as PNG"))

dl_layout.addWidget(QPushButton("Download as JPG"))
```

```
dl_layout.addWidget(QPushButton("Download as PDF"))

economic_layout.addLayout(dl_layout)


# View options

view_layout = QHBoxLayout()

view_layout.addWidget(QPushButton("View as Pie Chart"))

view_layout.addWidget(QPushButton("View as Table"))

economic_layout.addLayout(view_layout)


tabs.addTab(economic_tab, "Economic Cost")


# Social cost tab

social_tab = QWidget()

social_layout = QVBoxLayout(social_tab)

social_layout.addWidget(QLabel("Social cost distribution across stages for PSC bridges for
50 years"))

social_layout.addWidget(QWidget())  # Placeholder for chart

tabs.addTab(social_tab, "Social Cost")


# Environmental cost tab

env_tab = QWidget()

env_layout = QVBoxLayout(env_tab)

env_layout.addWidget(QLabel("Environmental cost distribution across stages for PSC
bridges for 50 years"))

env_layout.addWidget(QWidget())  # Placeholder for chart

tabs.addTab(env_tab, "Environmental Cost")


layout.addWidget(tabs)
```

```
class BLCCAStudio(QMainWindow):
    """Main BLCCA Studio application window"""

    def __init__(self):
        super(BLCCAStudio, self).__init__()
        self.initUI()

    def initUI(self):
        # Set window properties
        self.setWindowTitle("BLCCA Studio 1.0.0")
        self.setGeometry(100, 100, 1200, 800)

        # Create central widget
        central_widget = QWidget()
        self.setCentralWidget(central_widget)

        # Create menu bar
        self.createMenuBar()

        # Create toolbar
        self.createToolBar()

        # Create tab bar for main navigation
        self.createTopTabBar()

        # Create main layout
        main_layout = QHBoxLayout(central_widget)
```

```
main_layout.setContentsMargins(5, 5, 5, 5)
main_layout.setSpacing(0)

# Create splitter for resizable panels
splitter = QSplitter(Qt.Horizontal)

# Left panel - Project Details
project_panel = TreePanel()
project_panel.setMinimumWidth(250)
project_panel.setMaximumWidth(400)
# Add styling to match the left panel color
project_panel.setStyleSheet("background-color: #F5EFE0;")
splitter.addWidget(project_panel)

# Right side panel container
right_panel = QWidget()
right_layout = QVBoxLayout(right_panel)
right_layout.setContentsMargins(0, 0, 0, 0)
right_layout.setSpacing(0)

# Create tab widget for Data Window and Results Window
right_tabs = CloseableTabWidget()

# Data Window tab
data_tab = QScrollArea()
data_tab.setWidgetResizable(True)
data_tab.setWidget(DataWindowPanel())
right_tabs.addTab(data_tab, "Data Window")
```

```
# Results Window tab
results_tab = QScrollArea()
results_tab.setWidgetResizable(True)
results_tab.setWidget(ResultsWindowPanel())
right_tabs.addTab(results_tab, "Results Window")

right_layout.addWidget(right_tabs)
splitter.addWidget(right_panel)

# Set initial splitter sizes - left panel gets 1/4, right gets 3/4
splitter.setSizes([300, 900])

main_layout.addWidget(splitter)

# Set application style
self.setStyleSheet("""
    QMainWindow {
        background-color: #F0F0F0;
    }
    QTabWidget::pane {
        border: 1px solid #C0C0C0;
        background-color: white;
    }
    QTabBar::tab {
        background-color: #E0E0E0;
        border: 1px solid #C0C0C0;
        border-bottom: none;
```

```
            min-width: 8ex;
            padding: 5px;
        }
        QTabBar::tab:selected {
            background-color: white;
        }
        QPushButton {
            background-color: #F0F0F0;
            border: 1px solid #C0C0C0;
            border-radius: 3px;
            padding: 5px;
            min-width: 80px;
        }
        QPushButton:hover {
            background-color: #E0E0E0;
        }
        QSplitter::handle {
            background-color: #C0C0C0;
        }
        QTreeWidget {
            border: none;
        }
    """)

def createMenuBar(self):
    """Create the application menu bar"""
    menubar = self.menuBar()
```

```
    # File menu
    fileMenu = menubar.addMenu('File')
    fileMenu.addAction(self.createAction('New', 'Create a new project'))
    fileMenu.addAction(self.createAction('Open', 'Open an existing project'))
    fileMenu.addAction(self.createAction('Save', 'Save the current project'))
    fileMenu.addSeparator()
    fileMenu.addAction(self.createAction('Exit', 'Exit the application'))

    # Home menu
    homeMenu = menubar.addMenu('Home')

    # Reports menu
    reportsMenu = menubar.addMenu('Reports')

    # Help menu
    helpMenu = menubar.addMenu('Help')

def createToolBar(self):
    """Create the application toolbar"""
    toolbar = QToolBar("Main Toolbar")
    toolbar.setIconSize(QSize(16, 16))
    toolbar.setMovable(False)

    # Add blank placeholder icons for demonstration
    homeAction = self.createAction('', 'Home')
    fileAction = self.createAction('', 'Files')
    docAction = self.createAction('', 'Documents')
```

```
    toolbar.addAction(homeAction)
    toolbar.addAction(fileAction)
    toolbar.addAction(docAction)

    self.addToolBar(toolbar)

def createTopTabBar(self):
    """Create the top tab bar for main navigation"""
    tabBar = QTabWidget()
    tabBar.setDocumentMode(True)
    tabBar.setTabPosition(QTabWidget.North)

    # Create more centered navigation tabs with proper styling
    tabBar.setStyleSheet("""
        QTabBar {
            alignment: center;
        }
        QTabBar::tab {
            padding: 4px 16px;
            margin: 0;
            background-color: #F0F0F0;
            border: 1px solid #D0D0D0;
            border-bottom: none;
        }
        QTabBar::tab:selected {
            background-color: white;
        }
    """)
```

```
    # Create the tabs
    tabBar.addTab(QWidget(), "Windows")
    tabBar.addTab(QWidget(), "Tutorials")
    tabBar.addTab(QWidget(), "Project Details")
    tabBar.addTab(QWidget(), "Results")
    tabBar.addTab(QWidget(), "Compare")

    # Add tab bar below the toolbar with centered alignment
    tabBarContainer = QWidget()
    layout = QHBoxLayout(tabBarContainer)
    layout.setContentsMargins(0, 0, 0, 0)
    layout.setAlignment(Qt.AlignCenter)  # Center the tab widget in the container
    layout.addWidget(tabBar)

    # Create a dock widget to hold the tab bar
    dock = QDockWidget("", self)
    dock.setFeatures(QDockWidget.NoDockWidgetFeatures)
    dock.setWidget(tabBarContainer)
    dock.setTitleBarWidget(QWidget())  # Hide the title bar

    self.addDockWidget(Qt.TopDockWidgetArea, dock)

def createAction(self, text, statusTip):
    """Create a QAction with the given text and status tip"""
    action = QAction(text, self)
    action.setStatusTip(statusTip)
    return action
```

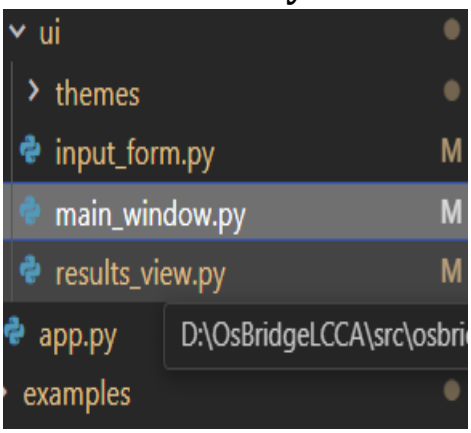
```
if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = BLCCAStudio()
    window.show()
    sys.exit(app.exec_())
```

3.3.3Explanation of the Code

- Imports PyQt5: For building GUI elements like menus, buttons, and layouts.
- Main Window (BICCAStudio): Sets title, size, and initializes the UI.
- Tutorial Panel: Displays 4 tutorial pages with navigation buttons.
- Menu Bar: Custom File and Help dropdowns with icons.
- Toolbar: Contains placeholder icons (doc, folder, print).
- Tabs: Adds tab buttons like Tutorials, Project Details, etc.
- Content Area: Split view with Tutorials (left) and Project Details (right).
- Project Details: Input fields for company, project info, valuer, etc.
- Status Bar: Shows a "Data" button.
- Event Handling: Hides open menus when clicking outside.

3.4 Task 1: Documentation

3.4.1 Directory Structure



Chapter 4

Internship Task 2 Title

4.1 Task 2: Problem Statement

To create a PyQt5-based GUI for **BLCCA Studio** that allows users to view and analyze 50-year bridge life-cycle cost data through interactive tabs, tree-structured project details, and visualizations like charts and tables.

4.2 Task 2: Tasks Done

Methodology:

Uses **modular GUI design** with PyQt5 to separate components (tree view, charts, tabs) for clarity and scalability.

Process:

1. Designed a main window layout using horizontal splitter (left: tree panel, right: results panel).

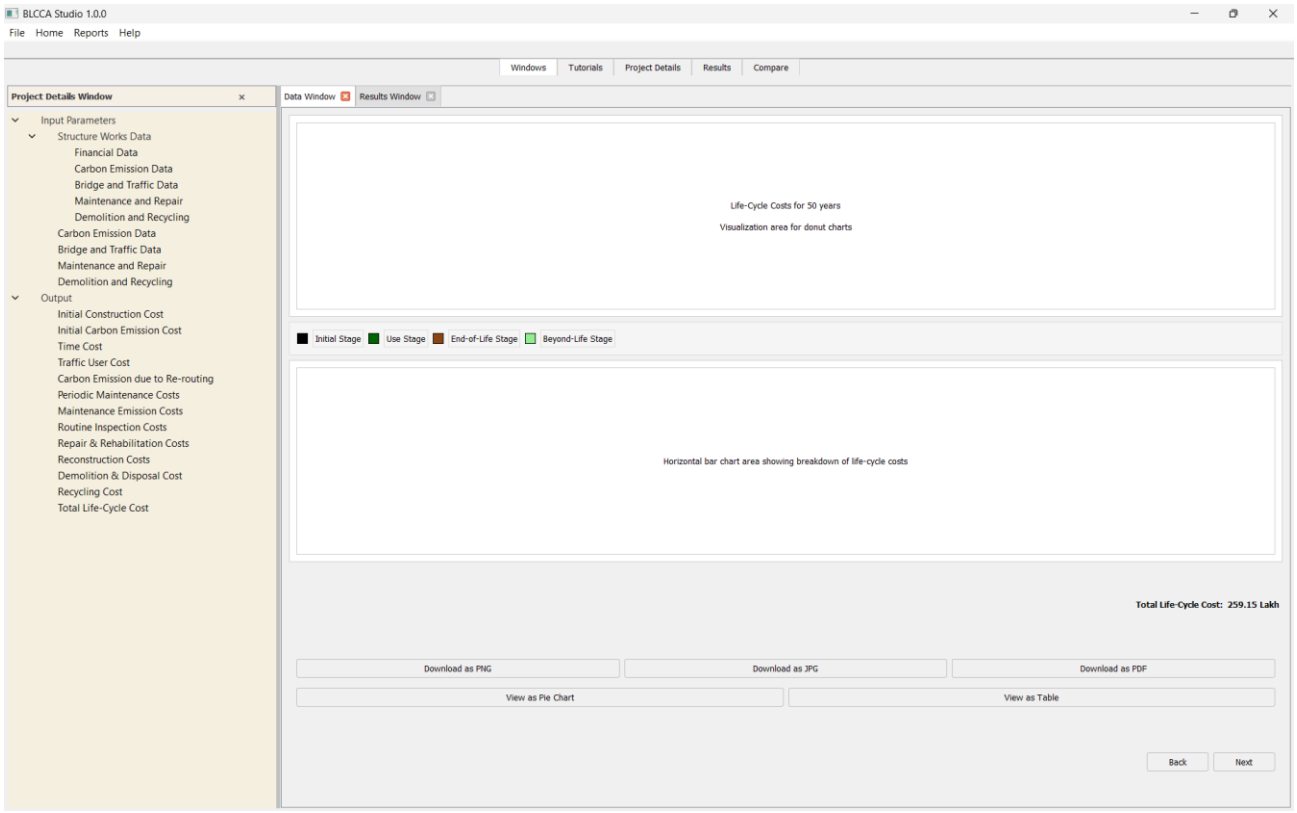
2. Implemented tabbed views to display different cost categories: Economic, Social, and Environmental.

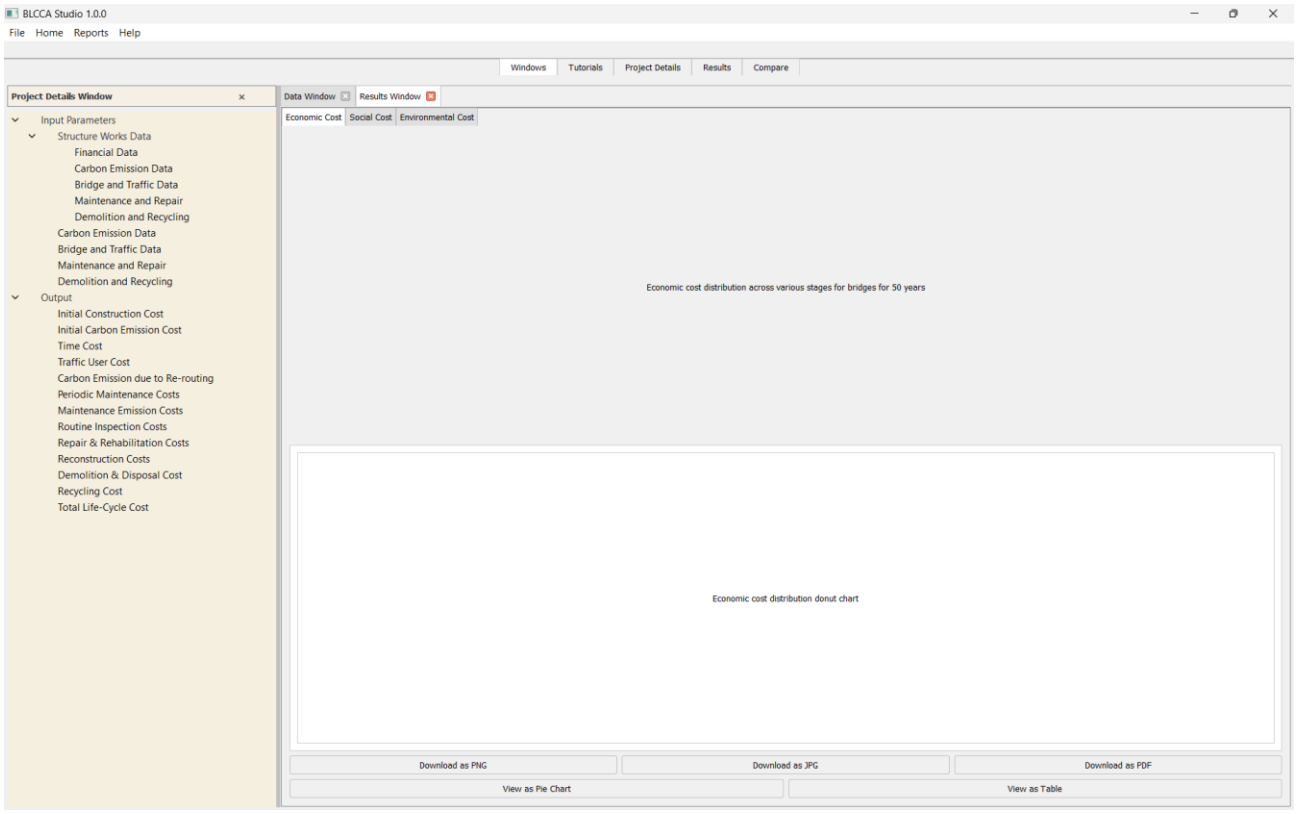
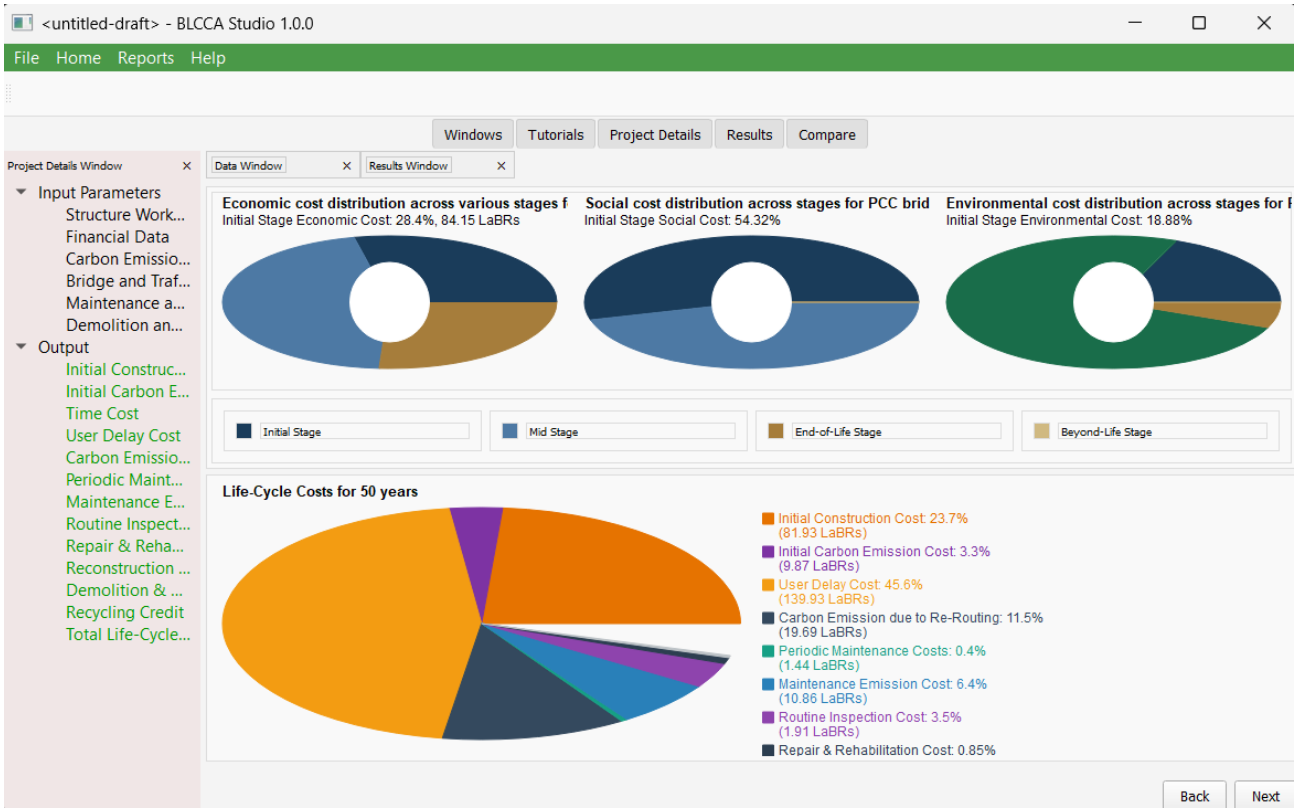
3. Visual areas created as placeholders for donut and bar charts with legends.

4. Added navigation, download (PNG, JPG, PDF), and view mode options (table/chart).

5. Used a QTreeWidget for structured input/output data navigation.

Implementation:

- **TreePanel**: Hierarchical project data (input/output parameters).
- **DataWindowPanel**: Cost visualizations and legends with download/view buttons.
- **ResultsWindowPanel**: Tabbed charts for cost categories.
- **BLCCAStudio**: Main application window integrating all components.
- 

- 
BLCCA Studio 1.0.0
File Home Reports Help
Windows Tutorials Project Details Results Compare
Project Details Window
Input Parameters
Structure Works Data
Financial Data
Carbon Emission Data
Bridge and Traffic Data
Maintenance and Repair
Demolition and Recycling
Carbon Emission Data
Bridge and Traffic Data
Maintenance and Repair
Demolition and Recycling
Output
Initial Construction Cost
Initial Carbon Emission Cost
Time Cost
Traffic User Cost
Carbon Emission due to Re-routing
Periodic Maintenance Costs
Maintenance Emission Costs
Routine Inspection Costs
Repair & Rehabilitation Costs
Reconstruction Costs
Demolition & Disposal Cost
Recycling Cost
Total Life-Cycle Cost
Data Window
Results Window
Economic Cost
Social Cost
Environmental Cost
Economic cost distribution across various stages for bridges for 50 years
Economic cost distribution donut chart
Download as PNG
Download as JPG
Download as PDF
View as Pie Chart
View as Table
- 
<untitled-draft> - BLCCA Studio 1.0.0
File Home Reports Help
Windows Tutorials Project Details Results Compare
Project Details Window
Input Parameters
Structure Work...
Financial Data
Carbon Emissio...
Bridge and Traf...
Maintenance a...
Demolition an...
Output
Initial Construc...
Initial Carbon E...
Time Cost
User Delay Cost
Carbon Emissio...
Periodic Maint...
Maintenance E...
Routine Inspect...
Repair & Reha...
Reconstruction ...
Demolition & ...
Recycling Credit
Total Life-Cycle...
Data Window
Results Window
Economic cost distribution across various stages f
Initial Stage Economic Cost: 28.4%, 84.15 LaBRs
Social cost distribution across stages for PCC brid
Initial Stage Social Cost: 54.32%
Environmental cost distribution across stages for
Initial Stage Environmental Cost: 18.88%
Initial Stage
Mid Stage
End-of-Life Stage
Beyond-Life Stage
Life-Cycle Costs for 50 years
Initial Construction Cost: 23.7%
(81.93 LaBRs)
Initial Carbon Emission Cost: 3.3%
(9.87 LaBRs)
User Delay Cost: 45.6%
(139.93 LaBRs)
Carbon Emission due to Re-Routing: 11.5%
(19.69 LaBRs)
Periodic Maintenance Costs: 0.4%
(1.44 LaBRs)
Maintenance Emission Cost: 6.4%
(10.86 LaBRs)
Routine Inspection Cost: 3.5%
(1.91 LaBRs)
Repair & Rehabilitation Cost: 0.85%
Back
Next

4.3 Full Code

```
import sys
from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QWidget, QVBoxLayout, QHBoxLayout,
    QLabel, QPushButton, QTabWidget, QTreeWidget, QTreeWidgetItem,
    QSplitter, QFrame, QSizePolicy, QAction, QToolBar, QStackedWidget,
    QDockWidget, QScrollArea, QMenu, QLineEdit, QGroupBox
)
from PyQt5.QtCore import Qt, QSize, QPoint
from PyQt5.QtGui import QIcon, QFont, QColor, QPalette, QPixmap


class CloseableTabWidget(QTabWidget):
    """Custom tab widget with closeable tabs"""

    def __init__(self, parent=None):
        super(CloseableTabWidget, self).__init__(parent)
        self.setTabsClosable(True)
        self.tabCloseRequested.connect(self.closeTab)
        self.setDocumentMode(True)
        self.setMovable(True)

    def closeTab(self, index):
        """Close the tab at the given index"""
        # Don't actually remove tabs in this demo
        print(f"Close tab requested for tab index {index}")

class TreePanel(QWidget):
```

```
"""Project details tree panel implementation"""

def __init__(self, parent=None):
    super(TreePanel, self).__init__(parent)
    self.initUI()

def initUI(self):
    layout = QVBoxLayout(self)
    layout.setContentsMargins(0, 0, 0, 0)

    # Create header
    header = QFrame()
    header.setFrameShape(QFrame.StyledPanel)
    # Updated header background color to match the image (light brown/beige)
    header.setStyleSheet("background-color: #F5EFE6;")
    header_layout = QHBoxLayout(header)
    header_layout.setContentsMargins(5, 5, 5, 5)

    title_label = QLabel("Project Details Window")
    title_label.setStyleSheet("color: #333333; font-weight: bold;")
    header_layout.addWidget(title_label)

    close_btn = QPushButton("×")
    close_btn.setFixedSize(20, 20)
    close_btn.setStyleSheet("""
        QPushButton {
            border: none;
```

```
        background-color: transparent;
        color: #555555;
        font-weight: bold;
        font-size: 14px;
    }
    QPushButton:hover {
        color: #000000;
    }
""")
header_layout.addWidget(close_btn)

layout.addWidget(header)

# Create tree widget
self.tree = QTreeWidget()
self.tree.setHeaderHidden(True)
self.tree.setStyleSheet("""
    QTreeWidget {
        border: none;
        background-color: #F5EFE0;  /* Beige background to match image */
    }
    QTreeWidget::item {
        height: 24px;
    }
    QTreeWidget::item:selected {
        background-color: #E6F2E6;
        color: #000000;
```

```
        }
        QTreeWidget::item:has-children {
            font-weight: bold;
            color: #333333;
        }
    """)

    # Input Parameters section
    input_params = QTreeWidgetItem(["Input Parameters"])
    input_params.setIcon(0, QIcon.fromTheme("folder", self.getDefaultIcon()))
    input_params.setBackground(0, QColor("#F5EFE0"))  # Set background for Input Parameters
    input_params.setForeground(0, QColor("#333333"))  # Dark text color
    self.tree.addTopLevelItem(input_params)

    # Structure Works Data
    structure_works = QTreeWidgetItem(["Structure Works Data"])
    structure_works.setIcon(0, QIcon.fromTheme("folder", self.getDefaultIcon()))
    structure_works.setForeground(0, QColor("#333333"))  # Dark text color
    input_params.addChild(structure_works)

    structure_works.addChild(self.createItem("Financial Data"))
    structure_works.addChild(self.createItem("Carbon Emission Data"))
    structure_works.addChild(self.createItem("Bridge and Traffic Data"))
    structure_works.addChild(self.createItem("Maintenance and Repair"))
    structure_works.addChild(self.createItem("Demolition and Recycling"))

    # Direct children of Input Parameters
```

```
input_params.addChild(self.createItem("Carbon Emission Data"))
input_params.addChild(self.createItem("Bridge and Traffic Data"))
input_params.addChild(self.createItem("Maintenance and Repair"))
input_params.addChild(self.createItem("Demolition and Recycling"))

# Output section
output = QTreeWidgetItem(["Output"])
output.setIcon(0, QIcon.fromTheme("folder", self.getDefaultIcon()))
output.setBackground(0, QColor("#F5EFE0"))  # Set background for Output
output.setForeground(0, QColor("#333333"))  # Dark text color
self.tree.addTopLevelItem(output)

# Output items
output_items = [
    "Initial Construction Cost",
    "Initial Carbon Emission Cost",
    "Time Cost",
    "Traffic User Cost",
    "Carbon Emission due to Re-routing",
    "Periodic Maintenance Costs",
    "Maintenance Emission Costs",
    "Routine Inspection Costs",
    "Repair & Rehabilitation Costs",
    "Reconstruction Costs",
    "Demolition & Disposal Cost",
    "Recycling Cost",
    "Total Life-Cycle Cost"
```

```
    ]

    for item_text in output_items:
        item = self.createItem(item_text, is_document=True)
        output.addChild(item)

    # Expand input params by default
    input_params.setExpanded(True)
    structure_works.setExpanded(True)
    output.setExpanded(True)

    layout.addWidget(self.tree)

def createItem(self, text, is_document=False):
    """Create a tree item with the appropriate icon"""
    item = QTreeWidgetItem([text])
    if is_document:
        # Document icon for output items
        item.setIcon(0, QIcon.fromTheme("text-x-generic", self.getDocumentIcon()))
    else:
        # Default folder icon
        item.setIcon(0, QIcon.fromTheme("folder", self.getDefaultIcon()))
    return item

def getDefaultIcon(self):
    """Return a default folder icon"""
    pixmap = QPixmap(16, 16)
```

```
        pixmap.fill(QColor(0, 0, 0, 0))
        return QIcon(pixmap)

    def getDocumentIcon(self):
        """Return a default document icon"""
        pixmap = QPixmap(16, 16)
        pixmap.fill(QColor(0, 0, 0, 0))
        return QIcon(pixmap)

class DataWindowPanel(QWidget):
    """Data Window Panel Implementation"""

    def __init__(self, parent=None):
        super(DataWindowPanel, self).__init__(parent)
        self.initUI()

    def initUI(self):
        layout = QVBoxLayout(self)

        # Placeholder for chart area - upper section
        chart_placeholder = QWidget()
        chart_placeholder.setMinimumHeight(300)
        chart_placeholder.setStyleSheet("background-color: #FFFFFF; border: 1px solid
#CCCCCC;")

        placeholder_text = QLabel("Life-Cycle Costs for 50 years\n\nVisualization area for donut
charts")
        placeholder_text.setAlignment(Qt.AlignCenter)
```

```
chart_layout = QVBoxLayout(chart_placeholder)
chart_layout.addWidget(placeholder_text)

layout.addWidget(chart_placeholder)

# Create legend
legend_frame = QFrame()
legend_frame.setFrameShape(QFrame.StyledPanel)
legend_frame.setMaximumHeight(50)
legend_frame.setStyleSheet("background-color: #F5F5F5; border: 1px solid #DDDDDD;")

legend_layout = QHBoxLayout(legend_frame)

legend_items = [
    ("Initial Stage", "#000000"),
    ("Use Stage", "#006400"),
    ("End-of-Life Stage", "#8B4513"),
    ("Beyond-Life Stage", "#90EE90")
]

for text, color in legend_items:
    color_box = QFrame()
    color_box.setFixedSize(16, 16)
    color_box.setStyleSheet(f"background-color: {color}; border: 1px solid black;")

    label = QLabel(text)
```

```
            item_layout = QHBoxLayout()
            item_layout.setContentsMargins(0, 0, 0, 0)
            item_layout.addWidget(color_box)
            item_layout.addWidget(label)

            legend_layout.addLayout(item_layout)

        legend_layout.addStretch()
        layout.addWidget(legend_frame)

        # Placeholder for horizontal bar chart - lower section
        barchart_placeholder = QWidget()
        barchart_placeholder.setMinimumHeight(300)
        barchart_placeholder.setStyleSheet("background-color: #FFFFFF; border: 1px solid
#CCCCCC;")

        barchart_text = QLabel("Horizontal bar chart area showing breakdown of life-cycle costs")
        barchart_text.setAlignment(Qt.AlignCenter)
        barchart_layout = QVBoxLayout(barchart_placeholder)
        barchart_layout.addWidget(barchart_text)

        layout.addWidget(barchart_placeholder)

        # Total cost display
        total_frame = QFrame()
        total_layout = QHBoxLayout(total_frame)
        total_layout.setContentsMargins(5, 5, 5, 5)
```

```
total_layout.addStretch()

total_label = QLabel("Total Life-Cycle Cost:")
total_label.setStyleSheet("font-weight: bold; font-size: 12px;")
total_layout.addWidget(total_label)

total_value = QLabel("259.15 Lakh")
total_value.setStyleSheet("font-weight: bold; font-size: 12px;")
total_layout.addWidget(total_value)

layout.addWidget(total_frame)

# Download and view options
options_frame = QFrame()
options_layout = QVBoxLayout(options_frame)

download_layout = QHBoxLayout()
download_layout.addWidget(QPushButton("Download as PNG"))
download_layout.addWidget(QPushButton("Download as JPG"))
download_layout.addWidget(QPushButton("Download as PDF"))
options_layout.addLayout(download_layout)

view_layout = QHBoxLayout()
view_layout.addWidget(QPushButton("View as Pie Chart"))
view_layout.addWidget(QPushButton("View as Table"))
options_layout.addLayout(view_layout)
```

```
        layout.addWidget(options_frame)

        # Navigation buttons
        nav_frame = QFrame()
        nav_layout = QHBoxLayout(nav_frame)

        nav_layout.addStretch()

        back_btn = QPushButton("Back")
        back_btn.setFixedWidth(100)
        nav_layout.addWidget(back_btn)

        next_btn = QPushButton("Next")
        next_btn.setFixedWidth(100)
        nav_layout.addWidget(next_btn)

        layout.addWidget(nav_frame)

class ResultsWindowPanel(QWidget):
    """Results Window Panel Implementation"""

    def __init__(self, parent=None):
        super(ResultsWindowPanel, self).__init__(parent)
        self.initUI()

    def initUI(self):
        layout = QVBoxLayout(self)
```

```
        layout.setContentsMargins(0, 0, 0, 0)


        # Create tabbed widget for Economic, Social, Environmental costs
        tabs = QTabWidget()
        tabs.setDocumentMode(True)


        # Economic cost tab
        economic_tab = QWidget()
        economic_layout = QVBoxLayout(economic_tab)


        cost_title = QLabel("Economic cost distribution across various stages for bridges for 50
years")
        cost_title.setAlignment(Qt.AlignCenter)
        economic_layout.addWidget(cost_title)


        chart_placeholder = QWidget()
        chart_placeholder.setMinimumHeight(300)
        chart_placeholder.setStyleSheet("background-color: #FFFFFF; border: 1px solid
#CCCCCC;")


        chart_text = QLabel("Economic cost distribution donut chart")
        chart_text.setAlignment(Qt.AlignCenter)
        chart_layout = QVBoxLayout(chart_placeholder)
        chart_layout.addWidget(chart_text)


        economic_layout.addWidget(chart_placeholder)


        # Download options
```

```
        dl_layout = QHBoxLayout()
        dl_layout.addWidget(QPushButton("Download as PNG"))
        dl_layout.addWidget(QPushButton("Download as JPG"))
        dl_layout.addWidget(QPushButton("Download as PDF"))
        economic_layout.addLayout(dl_layout)

        # View options
        view_layout = QHBoxLayout()
        view_layout.addWidget(QPushButton("View as Pie Chart"))
        view_layout.addWidget(QPushButton("View as Table"))
        economic_layout.addLayout(view_layout)

        tabs.addTab(economic_tab, "Economic Cost")

        # Social cost tab
        social_tab = QWidget()
        social_layout = QVBoxLayout(social_tab)
        social_layout.addWidget(QLabel("Social cost distribution across stages for PSC bridges for
50 years"))
        social_layout.addWidget(QWidget())  # Placeholder for chart
        tabs.addTab(social_tab, "Social Cost")

        # Environmental cost tab
        env_tab = QWidget()
        env_layout = QVBoxLayout(env_tab)
        env_layout.addWidget(QLabel("Environmental cost distribution across stages for PSC
bridges for 50 years"))
        env_layout.addWidget(QWidget())  # Placeholder for chart
```

```
        tabs.addTab(env_tab, "Environmental Cost")

        layout.addWidget(tabs)

class BLCCAStudio(QMainWindow):
    """Main BLCCA Studio application window"""

    def __init__(self):
        super(BLCCAStudio, self).__init__()
        self.initUI()

    def initUI(self):
        # Set window properties
        self.setWindowTitle("BLCCA Studio 1.0.0")
        self.setGeometry(100, 100, 1200, 800)

        # Create central widget
        central_widget = QWidget()
        self.setCentralWidget(central_widget)

        # Create menu bar
        self.createMenuBar()

        # Create toolbar
        self.createToolBar()

        # Create tab bar for main navigation
```

```
self.createTopTabBar()

# Create main layout
main_layout = QHBoxLayout(central_widget)
main_layout.setContentsMargins(5, 5, 5, 5)
main_layout.setSpacing(0)

# Create splitter for resizable panels
splitter = QSplitter(Qt.Horizontal)

# Left panel - Project Details
project_panel = TreePanel()
project_panel.setMinimumWidth(250)
project_panel.setMaximumWidth(400)
# Add styling to match the left panel color
project_panel.setStyleSheet("background-color: #F5EFE0;")
splitter.addWidget(project_panel)

# Right side panel container
right_panel = QWidget()
right_layout = QVBoxLayout(right_panel)
right_layout.setContentsMargins(0, 0, 0, 0)
right_layout.setSpacing(0)

# Create tab widget for Data Window and Results Window
right_tabs = CloseableTabWidget()
```

```
# Data Window tab
data_tab = QScrollArea()
data_tab.setWidgetResizable(True)
data_tab.setWidget(DataWindowPanel())
right_tabs.addTab(data_tab, "Data Window")

# Results Window tab
results_tab = QScrollArea()
results_tab.setWidgetResizable(True)
results_tab.setWidget(ResultsWindowPanel())
right_tabs.addTab(results_tab, "Results Window")

right_layout.addWidget(right_tabs)
splitter.addWidget(right_panel)

# Set initial splitter sizes - left panel gets 1/4, right gets 3/4
splitter.setSizes([300, 900])

main_layout.addWidget(splitter)

# Set application style
self.setStyleSheet("""
    QMainWindow {
        background-color: #F0F0F0;
    }
    QTabWidget::pane {
        border: 1px solid #C0C0C0;
```

```
    background-color: white;
}
QTabBar::tab {
    background-color: #E0E0E0;
    border: 1px solid #C0C0C0;
    border-bottom: none;
    min-width: 8ex;
    padding: 5px;
}
QTabBar::tab:selected {
    background-color: white;
}
QPushButton {
    background-color: #F0F0F0;
    border: 1px solid #C0C0C0;
    border-radius: 3px;
    padding: 5px;
    min-width: 80px;
}
QPushButton:hover {
    background-color: #E0E0E0;
}
QSplitter::handle {
    background-color: #C0C0C0;
}
QTreeWidget {
    border: none;
```

```
        }
    """)

def createMenuBar(self):
    """Create the application menu bar"""
    menubar = self.menuBar()

    # File menu
    fileMenu = menubar.addMenu('File')
    fileMenu.addAction(self.createAction('New', 'Create a new project'))
    fileMenu.addAction(self.createAction('Open', 'Open an existing project'))
    fileMenu.addAction(self.createAction('Save', 'Save the current project'))
    fileMenu.addSeparator()
    fileMenu.addAction(self.createAction('Exit', 'Exit the application'))

    # Home menu
    homeMenu = menubar.addMenu('Home')

    # Reports menu
    reportsMenu = menubar.addMenu('Reports')

    # Help menu
    helpMenu = menubar.addMenu('Help')

def createToolBar(self):
    """Create the application toolbar"""
    toolbar = QToolBar("Main Toolbar")
```

```
    toolbar.setIconSize(QSize(16, 16))
    toolbar.setMovable(False)

    # Add blank placeholder icons for demonstration
    homeAction = self.createAction('', 'Home')
    fileAction = self.createAction('', 'Files')
    docAction = self.createAction('', 'Documents')

    toolbar.addAction(homeAction)
    toolbar.addAction(fileAction)
    toolbar.addAction(docAction)

    self.addToolBar(toolbar)

def createTopTabBar(self):
    """Create the top tab bar for main navigation"""
    tabBar = QTabWidget()
    tabBar.setDocumentMode(True)
    tabBar.setTabPosition(QTabWidget.North)

    # Create more centered navigation tabs with proper styling
    tabBar.setStyleSheet("""
        QTabBar {
            alignment: center;
        }
        QTabBar::tab {
            padding: 4px 16px;
```

```
        margin: 0;
        background-color: #F0F0F0;
        border: 1px solid #D0D0D0;
        border-bottom: none;
    }
    QTabBar::tab:selected {
        background-color: white;
    }
""")

# Create the tabs
tabBar.addTab(QWidget(), "Windows")
tabBar.addTab(QWidget(), "Tutorials")
tabBar.addTab(QWidget(), "Project Details")
tabBar.addTab(QWidget(), "Results")
tabBar.addTab(QWidget(), "Compare")

# Add tab bar below the toolbar with centered alignment
tabBarContainer = QWidget()
layout = QHBoxLayout(tabBarContainer)
layout.setContentsMargins(0, 0, 0, 0)
layout.setAlignment(Qt.AlignCenter)  # Center the tab widget in the container
layout.addWidget(tabBar)

# Create a dock widget to hold the tab bar
dock = QDockWidget("", self)
dock.setFeatures(QDockWidget.NoDockWidgetFeatures)
```

```
        dock.setWidget(tabBarContainer)
        dock.setTitleBarWidget(QWidget())  # Hide the title bar

        self.addDockWidget(Qt.TopDockWidgetArea, dock)

    def createAction(self, text, statusTip):
        """Create a QAction with the given text and status tip"""
        action = QAction(text, self)
        action.setStatusTip(statusTip)
        return action

if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = BLCCAStudio()
    window.show()
    sys.exit(app.exec_())
```

Chapter 5

Conclusions

5.1 TasksAccomplished

During my internship, I successfully designed and developed the **user interface** for the desktop application, including the **main window** and **results window**. This involved creating structured layouts, tab navigation, input/output sections, and placeholders for data visualization using **PyQt5**, ensuring a clean, responsive, and user-friendly GUI.

5.2 Skills Developed

PyQt5 GUI Development – Gained hands-on experience in designing responsive and modular desktop interfaces.

UI/UX Design Principles – Learned to create user-friendly layouts with clear navigation and visual hierarchy.

Object-Oriented Programming (OOP) – Improved understanding of class-based architecture for scalable code.

Layout & Widget Management – Worked with layouts like QVBoxLayout, QHBoxLayout, QSplitter, and various widgets.

Event Handling – Implemented interactivity using PyQt5 signals and slots (e.g., button clicks, tab switching).

Project Structuring – Organized GUI components into reusable and maintainable code modules.

Chapter A

Appendix

A.1 Work Reports

Internship Work Report			
Name:		Om Lakshkar	
Project:		Development of UI for OsBLCCA Tool for Osdag	
Internship:		FOSSEE Semester long Internship program	
DATE	**DAY**	**TASK**	**Hours Worked**
17-feb-2025	Monday	Installed OSDAG on Windows 10 x64 machine and did initial testing	4
20-feb-2025	Thursday	Continued testing OSDAG on Windows 10 machine and validated installation process	4
25-feb-2025	Tuesday	Gone through the Osdag repo	5
28-feb-2025	Friday	Understanding the Osdag codebase	4
4-march-2025	Tuesday	Did fixes in setting up the repo	4
15-march-2025	Saturday	LCC Tool setup	2
19-march-2025	Wednesday	Exam Break	0
20-march-2025	Thursday	Exam Break	0
21 march 2025	Friday	Exam Break	0
22 march 2025	Saturday	Exam break	6
27 march 2025	Thursday	Meeting with mentor	1
29 march 2025	Saturday	Meeting with mentor for further discussion on desktop app	1
5 april 2025	Saturday	Understanding the documentation of desktop app	5
8 april 2025	Tuesday	Started working on ui of desktop app	6
20 april 2025	sunday	Started working on other pages	5
30 april 2025	wednesday	Meeting with team	1
2 may 2025	Friday	Started working on result window	7
5 may 2025	monday	Meeting with team	1
10 may 2025	Saturday	Exam Break	0
12 may 2025	Monday	Exam Break	0

20 may 2025	Tuesday	Completed the result window work	6
21 may 2025	Wednesday	Started working on other pages of ui	7
23 may 2025	friday	Fixes some issue in the ui	6
25 may 2025	sunday	Completed the majority portion of ui	6
29 may 2025	Thursday	Meeting with team	1
31 may 2025	Saturday	Final completion of the project	6

Bibliography

[1] Siddhartha Ghosh, Danish Ansari, Ajmal Babu Mahasrankintakam, Dharma Teja Nuli, Reshma Konjari, M. Swathi, and Subhrajit Dutta.

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.

[3] FOSSEE Project. Osdag website.