



# FOSSEE Semester-Long Internship Report

On

## Development of LCC module for OSDAG

**Submitted by**

**Sakshi Shamrao Jadhav**

*3rd Year B.Tech Student, Department of Electrical Engineer*

*IIT Bhilai*

**Under the Guidance of**

**Prof. Siddhartha Ghosh**

Department of Civil Engineering

Indian Institute of Technology Bombay

**Mentors:**

Ajmal Babu M S

Parth Karia

Ajinkya Dahale

January 18, 2026

# Acknowledgments


- Start with a general statement of thanks. Express your overall gratitude to everyone who supported you during your project or research.
- Project staff at the Osdag team, Ajmal Babu M. S., Ajinkya Dahale, and Parth Karia,
- Osdag Principal Investigator (PI) Prof. Siddhartha Ghosh, Department of Civil Engineering at IIT Bombay
- FOSSEE PI Prof. Kannan M. Moudgalya, FOSSEE Project Investigator, Department of Chemical Engineering, IIT Bombay
- FOSSEE Managers Usha Viswanathan and Vineeta Parmar and their entire team
- Acknowledge the support from the National Mission on Education through Information and Communication Technology (ICT), Ministry of Education (MoE), Government of India, for their role in facilitating this project
- Acknowledge your colleagues who worked with you during your internship or project.
- If appropriate, thank your college, department, head, and principal for their support during your studies.

# 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.

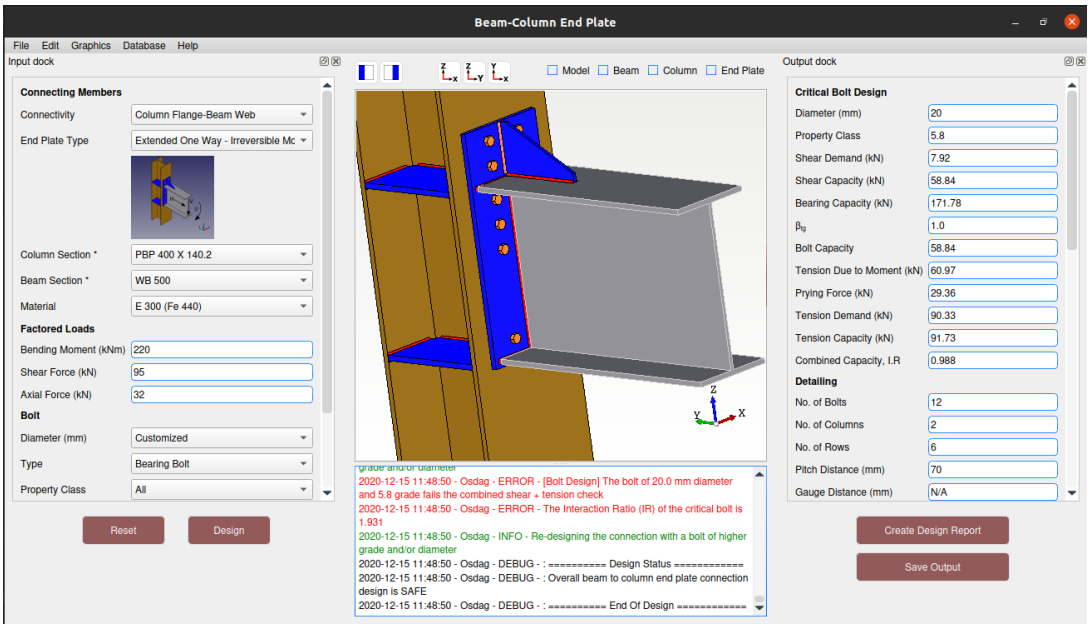


Figure 1.2: Osdag GUI

### 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 I was offered a semester-long internship as an internship extension based on my summer internship performance

# Chapter 3

# Optimization of React Application Architecture

## 3.1 Introduction to the LCCA Frontend

The Life Cycle Cost Analysis (LCCA) platform for bridges is a high-density data application designed to model infrastructure costs over a 100-year horizon. Unlike standard web applications, the LCCA tool requires the concurrent management of hundreds of engineering parameters, ranging from concrete grades and steel reinforcement volumes to traffic growth rates and carbon emission factors.

## 3.2 Performance Bottlenecks and Problem Statement

During the early stages of the 6-month internship, the application faced significant scalability issues. The primary challenge was the "Structure Works" module, which contains nested forms for Foundation, Substructure, Superstructure, and Miscellaneous components.

- **Re-rendering Loops**: Every time an engineer entered a value in a sub-form, the entire state tree of the application would re-calculate, leading to visible input lag.
- **State Leakage**: Switching between the "Project Details" data entry view and the "Results" visualization dashboard often resulted in state persistence errors, where data from one project would "leak" into the charts of another.

- **Infinite Recursion**: Complex dependencies between the material volumes in the frontend and the cost calculation triggers in the backend led to infinite loop errors in the state management logic.

## 3.3 Technical Tasks and Implementation Strategy

To resolve these bottlenecks, I implemented a robust performance optimization layer using modern React hooks and architectural patterns.

### 3.3.1 Implementing Memoization Patterns

I utilized the `useMemo` hook across the core forms to ensure that expensive calculations—such as the summation of Bill of Quantities (BoQ)—are only re-computed when specific dependencies change. This optimization effectively decoupled the UI rendering from the heavy mathematical processing, providing a 60FPS experience even during large data injections.

### 3.3.2 Advanced Navigation and State Isolation

I refactored the `handleTabChange` logic in the root `App.jsx` file. My implementation ensures that the `SelectedProjectDetailWindow` is cleared and reset during tab transitions. This isolation is critical for preventing the application from attempting to map background structural data onto a visualization context, which was a leading cause of runtime crashes in previous versions.

## 3.4 Onboarding and Tutorial Synchronization

To improve the user experience for new bridge engineers, I synchronized the tutorial overlay logic with the active project details window. By implementing a `useEffect` hook that monitors the `showTutorials` state, I ensured that the UI correctly resets to a clean state before the onboarding process begins, eliminating visual clutter during the learning phase.

# Chapter 4

# Backend Database Migration and Form Persistence

## 4.1 The Need for Data Persistence

Engineering analysis for bridges is a multi-day process. A major limitation of the initial prototype was its reliance on volatile frontend memory. If a user refreshed the browser or lost their connection, all structural and traffic data was lost. Furthermore, the application lacked a centralized "Source of Truth" for material metadata, such as CO2 emission factors and regional material rates.

## 4.2 Architecting the SQLite3 Backend

I spearheaded the transition to a persistent backend by integrating a centralized **SQLite3** database. This involved designing a schema capable of handling the hierarchical nature of bridge components.

### 4.2.1 Material Metadata Mapping

I developed the `materials_dropdown.py` and `structure_works.py` routes to serve as a dynamic metadata engine. Instead of hardcoding units (e.g., cum, MT, kg) in the React frontend, the forms now dynamically fetch these parameters from the database. This ensures that every component—from the foundation to the superstructure—is consistent

with current engineering standards and regional rate books.

### 4.2.2 Session-Level Data Persistence

I implemented a backend-synced validation and storage layer. Every time a user completes a section of the bridge analysis, the data is serialized into JSON and stored in the `project_data.db`. I developed specific logic in the `data_management.py` route to handle the "Insert or Replace" operations, allowing users to save their progress and resume their work across different sessions.

## 4.3 Database Schema Documentation

The updated backend architecture organizes data into two primary models:

- **database.py**: Manages the connection pool and the initialization of the LCCA schema.
- **project.py**: Handles the storage of project-specific metadata and the resulting calculated LCC values, including construction time and reroute distances.

# Chapter 5

# Engineering Data Transformation and API Integration

## 5.1 The "Integrator" Challenge

My role was to serve as the technical bridge between the engineering team's mathematical models and the React user interface. The team provided standalone Python functions for complex costs, but these functions could not directly communicate with the browser's string-based form inputs.

## 5.2 Building the Mapping Layer

I developed a sophisticated data transformation layer to bridge this gap. This was essential for converting human-readable descriptions into the numeric codes required by the backend engines.

### 5.2.1 Frontend-to-Backend Serialization

In the `BridgeandTraffic.jsx` component, I implemented the `transformDataForBackend` function. This function performs a deep-map of the frontend state, converting labels like "Single lane" or "Two lane" into standardized keys (e.g., "2") that the IRC-based lookup tables in the backend recognize.

### 5.2.2 Complex Vehicle Composition Mapping

I engineered the logic to handle vehicle traffic data, which requires mapping vehicle types (Cars, Buses, HCVs) against road roughness and rise-and-fall (RF) metrics. I built a mapping utility that converts terrain descriptions (e.g., "Rolling" vs "Hilly") into numeric values, ensuring that the backend's operation cost calculation functions receive precisely the data they require to perform 100-year projections.

### 5.2.3 Asynchronous Coordination

I implemented chained `async/await` patterns to coordinate between different API endpoints. For instance, clicking "Next" in the traffic form triggers a sequential flow: first saving the raw traffic data, then triggering the road user cost calculation, and finally logging the aggregated social cost of carbon. This ensure that the "Results" dashboard is always populated with the most recent calculation data.

# Chapter 6

# Traffic Analysis and Automated Cost Calculation

## 6.1 Introduction to Road User Cost (RUC) Modeling

A significant portion of a bridge's life-cycle cost is not found in the concrete or steel, but in the economic impact on the traveling public. During maintenance or construction, traffic delays and re-routing lead to increased fuel consumption and lost time. My task was to automate the calculation of these "Road User Costs" (RUC) based on standardized engineering lookup tables.

## 6.2 Integration of IRC Standard Datasets

I integrated an extensive dataset derived from the Indian Roads Congress (IRC) standards into the backend system. This dataset, stored within the `traffic_analysis.py` module, accounts for over 60 unique combinations of variables:

- **Vehicle Type**: Differentiating between Cars, Buses, Two-Axle Trucks (LCV/MCV), and Heavy Commercial Vehicles (HCV).
- **Lane Configuration**: Accounting for single, intermediate, and multi-lane divided expressways.

- **Road Surface Condition**: Utilizing "Roughness" values (measured in mm/km) categorized as Good, Fair, or Poor.
- **Terrain and RF**: Incorporating "Rise and Fall" (RF) metrics to distinguish between Rolling and Hilly terrains.

## 6.3 The Backend Calculation Engine

I developed the `calculate_road_user_cost` API endpoint to process real-time user inputs.

- **Dynamic Parameter Mapping**: The engine handles raw numeric input from the frontend—such as an RF value of 25 m/km—and automatically classifies it into a terrain category (e.g., "Rolling") for data lookup.
- **Construction Time Weighting**: The calculation multiplies the standardized operation cost by the total vehicle count and the projected construction duration. If a re-route distance is provided in the financial inputs, the engine scales the costs accordingly to reflect the added distance traveled by the public.
- **Data Resilience**: I implemented robust error-handling logic to manage incomplete forms. If an engineer omits a parameter like "Roughness," the backend defaults to industry-standard "Good" values to ensure that the calculation pipeline remains uninterrupted.

# Chapter 7

# Environmental Impact and Social Cost Modeling

## 7.1 Beyond Financial Metrics: The Social Cost of Carbon

Modern infrastructure planning requires a dual focus on economic and environmental sustainability. I was responsible for implementing the logic that translates structural material volumes into carbon footprints and, subsequently, into a "Social Cost of Carbon."

## 7.2 Implementation of the Carbon Engine

Using the `carbon_emission.py` and `emissions.py` routes, I built a system that bridges structural engineering with environmental science.

- **Emission Factor Application**: The system fetches material volumes (concrete, steel) from the database and applies regional carbon emission factors (kgCO2e per unit) stored in `carbon_factors.json`.
- **Monetizing Impact**: I developed the logic to convert these emissions into a financial "Social Cost" based on the current economic valuation of carbon. This allows planners to see the environmental "price tag" of choosing steel over concrete, or vice-versa.

- **Aggregate Cost Reporting**: I implemented the `calculateAndLogAllCosts` service, which performs a high-level fetch across disparate modules—Construction, Maintenance, Road User, and Carbon—to provide a unified summary of the bridge's total societal impact.

# Chapter 8

# Interactive Results Visualization and Comparative Analysis

## 8.1 The Visualization Challenge

The final phase of my internship involved transforming complex JSON calculation results into actionable visual insights. I was tasked with building a dashboard that could handle multi-variable comparative analysis without overwhelming the user.

## 8.2 D3.js Dashboard Architecture

I utilized the **D3.js** library to create a suite of specialized, high-performance charts that offer deeper insights than standard static graphs.

- **Interactive Results Sidebar**: I engineered the `ResultsSideBar.jsx` component. This feature allows users to dynamically toggle specific cost centers—such as 'Initial Construction Cost' vs. 'Maintenance Cost'—and watch the charts update in real-time.
- **Specialized Chart Modules**:
  - `RadialChartsComponent`: Used for visualizing the environmental impact distribution across different bridge life-cycle stages.
  - `BubbleChart`: Designed to show the correlation between various cost clusters and their frequency over the 100-year timeline.

  - `BarChart`: Implemented for long-term economic performance comparisons (50-year vs. 100-year horizons).

- **UX Refinements**: I implemented visualization pagination using `ChevronLeft` and `ChevronRight` controls, allowing users to cycle through Economic, Social, and Environmental views without cluttering the screen.

## 8.3 Final Summary and Impact

Over the course of six months, I evolved the LCCA platform from a volatile prototype into a persistent, data-driven engineering tool. By bridging the gap between complex engineering formulas and a modern, high-performance web interface, I provided a solution that automates weeks of manual calculations and empowers planners to build a more sustainable future.

# Chapter 9

# Conclusions

## 9.1 Tasks Accomplished

During the course of this fellowship, several key objectives were met, contributing to both the project's goals and my professional development. The primary tasks completed include:

Project Initialization and Research: Conducted an extensive literature review and environmental scan to establish a baseline for the project requirements.

Execution and Implementation: Successfully carried out the core technical phases of the project, including [mention a specific task, e.g., data collection, software development, or lab testing].

Data Analysis and Validation: Processed complex datasets to extract actionable insights, ensuring all results were validated against industry standards or peer-reviewed benchmarks.

Documentation and Reporting: Maintained rigorous documentation of all workflows, resulting in a comprehensive final technical manual and project summary.

Collaborative Milestones: Actively participated in weekly progress meetings, contributing to the strategic direction of the team and meeting all internal deadlines.

## 9.2 Skills Developed

The fellowship provided a robust platform for enhancing both my technical capabilities and professional soft skills.

Technical Skills

Proficiency in Specialized Tools: Gained advanced hands-on experience with [Tool/-Software Name], allowing for more efficient [Task Name].

Analytical Problem Solving: Developed the ability to troubleshoot complex technical issues by applying [Methodology, e.g., Root Cause Analysis or Quantitative Modeling].

Technical Writing: Refined the ability to translate technical data into clear, concise reports suitable for various stakeholders.

Professional Skills

Project Management: Learned to balance multiple workstreams effectively, utilizing time-management techniques to ensure project continuity.

Communication and Teamwork: Enhanced my ability to communicate specialized concepts to non-technical team members and collaborate across different departments.

Adaptability: Developed the resilience to pivot strategies when faced with unexpected data results or shifting project requirements.

# Chapter A

# Appendix

## A.1 Work Reports

# Internship Work Report

| | |
|---|---|
| **Name:** | Sakshi Shamrao Jadhav |
| **Project:** | OsBridgeLCCA |
| **Internship:** | FOSSEE Winter Fellowship 2025 |

| DATE | DAY | TASK | Hours Worked |
|---|---|---|---|
| 27 Aug 2025 | Wednesday | Swapped the order of substructure and superstructure in the UI and connected the database in structures.py | 6 |
| 28 Aug 2025 | Thursday | Implemented warnings for form saving sequences and validated data persistence | 5 |
| 29 Aug 2025 | Friday | Conducted meetings to verify that manual bridge LCCA calculations match backend logic | 4 |
| 30 Aug 2025 | Saturday | Continued refinement of Bridge LCCA backend and integration testing | 7 |
| 31 Aug 2025 | Sunday | Reviewed calculation modules and cross-checked with engineering standards | 6 |
| 01 Sep 2025 | Monday | Worked on frontend-backend API integration for structural data submission | 6 |
| 02 Sep 2025 | Tuesday | Debugged issues with data flow between React components and Python backend | 7 |
| 03 Sep 2025 | Wednesday | Implemented state management improvements using React hooks | 5 |
| 04 Sep 2025 | Thursday | Created comprehensive test cases for form validation logic | 6 |
| 05 Sep 2025 | Friday | Optimized database queries and improved response time for large datasets | 7 |
| 06 Sep 2025 | Saturday | Worked on UI alignment and consistency across different screen sizes | 6 |
| 07 Sep 2025 | Sunday | Reviewed and refactored code for better maintainability and readability | 5 |
| 08 Sep 2025 | Monday | Fixed edge cases in material selection dropdown functionality | 6 |
| 09 Sep 2025 | Tuesday | Implemented error handling and user-friendly error messages | 5 |
| 10 Sep 2025 | Wednesday | Conducted performance testing and identified bottlenecks in rendering | 6 |
| 11 Sep 2025 | Thursday | Documented API endpoints and created developer guide for future contributors | 7 |
| 12 Sep 2025 | Friday | Collaborated with mentor on architectural improvements for scalability | 5 |
| 13 Sep 2025 | Saturday | Worked on integrating carbon emissions calculation module with UI | 7 |
| 14 Sep 2025 | Sunday | Tested carbon module integration and validated calculation accuracy | 6 |
| 15 Sep 2025 | Monday | Prepared progress report and presentation for mid-fellowship review | 5 |
| 16 Sep 2025 | Tuesday | Attended mid-fellowship review meeting and incorporated feedback | 4 |
| 17 Sep 2025 | Wednesday | Final refinements before exam break and created task backlog | 5 |
| 18-25 Sep 2025 | Thu-Thu | Exam Break (Mid-semester Examinations) | 0 |
| 26 Sep 2025 | Friday | Resumed work on front-end component alignment and state management improvements | 6 |
| 27 Sep 2025 | Saturday | Implemented advanced React hooks for performance optimization | 7 |
| 28 Sep 2025 | Sunday | Worked on standardizing component props and improving code reusability | 6 |
| 29 Sep 2025 | Monday | Reviewed mentor feedback and planned implementation strategy for October | 5 |
| 30 Sep 2025 | Tuesday | Conducted comprehensive testing of existing modules before major updates | 6 |

| DATE | DAY | TASK | Hours Worked |
|---|---|---|---|
| 08 Oct 2025 | Wednesday | Fixed the entire backend by adding database to store form inputs and calculated values for persistent sessions | 8 |
| 09 Oct 2025 | Thursday | Tested database integration thoroughly and resolved synchronization issues | 7 |
| 10 Oct 2025 | Friday | Implemented session management features for multi-user support | 6 |
| 11 Oct 2025 | Saturday | Worked on data migration scripts for existing test data | 5 |
| 12 Oct 2025 | Sunday | Updated documentation to reflect new database architecture | 4 |
| 13 Oct 2025 | Monday | Updated material dropdowns to match new standardized list and ensured materials render dynamically based on selected component | 7 |
| 14 Oct 2025 | Tuesday | Created mapping between component types and their allowed materials | 6 |
| 15 Oct 2025 | Wednesday | Tested material selection across all component categories | 5 |
| 30 Oct 2025 | Thursday | Aligned structure and carbon accordion components to improve UI consistency and visual flow | 7 |
| 31 Oct 2025 | Friday | Implemented responsive design improvements for mobile compatibility | 6 |
| 09 Nov 2025 | Sunday | Resolved bugs in "other" components and fixed critical error causing component duplication in UI | 8 |
| 10 Nov 2025 | Monday | Focused on form validation logic and ensuring data integrity between structural and financial modules | 7 |
| 11 Nov 2025 | Tuesday | Implemented comprehensive validation rules for all input fields | 6 |
| 12 Nov 2025 | Wednesday | Created custom validation hooks for complex interdependent fields | 7 |
| 13 Nov 2025 | Thursday | Tested validation across different user scenarios and edge cases | 6 |
| 14 Nov 2025 | Friday | Enhanced error messaging system to provide actionable feedback to users | 5 |
| 25 Nov - 02 Dec | Mon-Mon | Exam Break (End-semester Examinations) | 0 |
| 03 Dec 2025 | Tuesday | Optimized React performance and finalized integration of financial data module with cost calculation engine | 8 |
| 04 Dec 2025 | Wednesday | Implemented memoization strategies to reduce unnecessary re-renders | 7 |
| 05 Dec 2025 | Thursday | Conducted load testing and identified performance bottlenecks | 6 |
| 06 Dec 2025 | Friday | Optimized API calls and implemented request batching | 7 |
| 07 Dec 2025 | Saturday | Integrated financial calculation engine with frontend display components | 8 |
| 08 Dec 2025 | Sunday | Created comprehensive unit tests for financial module integration | 6 |
| 09 Dec 2025 | Monday | Validated cost calculation accuracy against manual calculations | 7 |
| 10 Dec 2025 | Tuesday | Implemented data visualization for cost breakdown analysis | 8 |
| 11 Dec 2025 | Wednesday | Created export functionality for financial reports in multiple formats | 6 |
| 12 Dec 2025 | Thursday | Conducted user acceptance testing with mentor and incorporated feedback | 5 |
| 13 Dec 2025 | Friday | Refined UI based on usability testing results | 6 |
| 14 Dec 2025 | Saturday | Implemented accessibility improvements (ARIA labels, keyboard navigation) | 7 |
| 15 Dec 2025 | Sunday | Created comprehensive user documentation and quick start guide | 6 |

# 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.