FOSSEE Semester Long Internship 2026

On

Web version of 3pcLCCABridge

Submitted by

Raj Verma

4th Year B.Tech Student, Department of computer Science and Engineering

Vellore Institute of Technology, Bhopal University

Mumbai

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

May 27, 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.

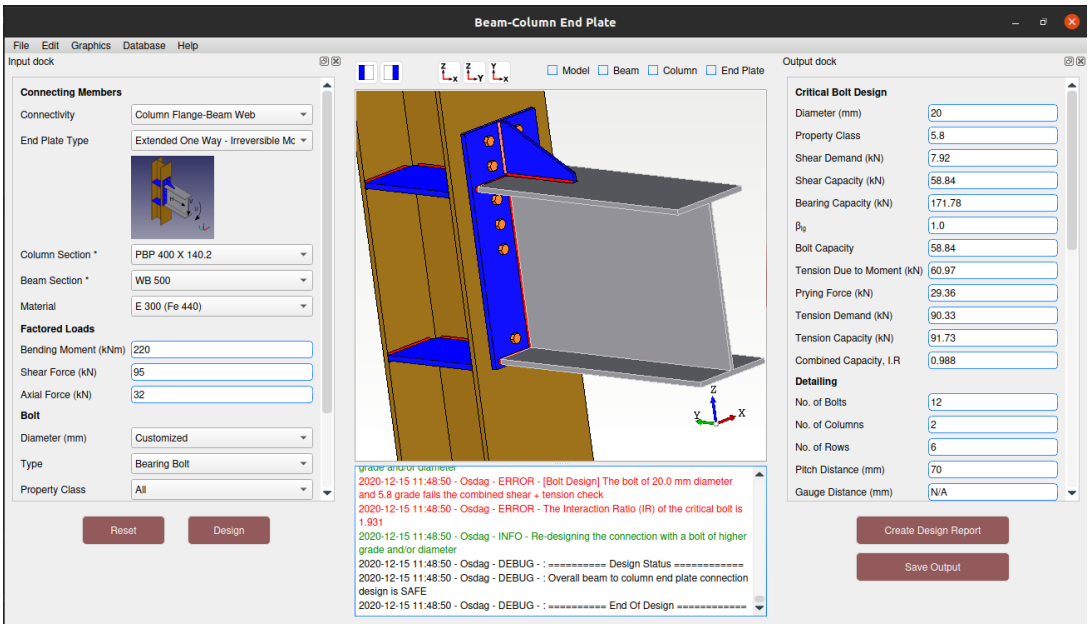


Figure 1.2: Osdag GUI

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 Problem Statement

The legacy OSDAG (Open Steel Design and Graphics) LCCA desktop application suffered from multiple UI layout inconsistencies that hindered usability. Specifically, resizing the desktop window led to widget overlapping, text label clipping, and misaligned unit fields.

Additionally, the desktop tool lacked a structured cloud backup mechanism. Users had no automated method to secure their bridge lifecycle data against accidental loss, necessitating the design of an integrated Google Drive backup protocol that could securely authenticate and sync local project state files (`.3psLCCAFile`) to a remote cloud directory.

2.2 Tasks Done

To resolve these issues, the following engineering tasks were successfully performed:

1. **PySide6 UI Layout Refactoring**:
    - Replaced absolute widget positioning with dynamic layout managers (`QVBoxLayout`, `QHBoxLayout`, and `QGridLayout`) in PySide6 to ensure fluid responsiveness during window scaling.
    - Adjusted grid spacing, padding, and layout alignments to prevent text clipping and secure clean, standard unit input alignments.
2. **Google Drive Backup Syncing Protocol**:

- Designed and scripted a secure authentication flow leveraging Google OAuth 2.0 client credentials.
- Developed Python routines using the Google Drive v3 API to detect changes in local project files and perform secure, automated incremental uploads to a dedicated user backup folder in the cloud.

Chapter 3

React UI Shell & Carbon Emissions Container

3.1 Task 1: Problem Statement

The legacy FOSSEE 3psLCCA application was written as a desktop interface using Python and PySide6 (Qt). While fully operational, it required local environment setups, compilation of OCC libraries, and manual installation packages, creating accessibility barriers for engineers, students, and collaborators.

The first major task of the internship was to port the core desktop user interface to a clean, modular React and Bootstrap web application. The core challenges included:

- Recreating a deeply nested collapsible sidebar tree structure that mimics the PySide6 sidebar navigation layout.
- Establishing a unified page layout container that coordinates the top navbar controls, sidebar menus, and form widgets.
- Modernizing the Carbon Emissions container to provide edge-to-edge centralized scrolling, dual-mode lump-sum/detailed forms, and sticky real-time summary calculators.
- Developing a real-time system log console to stream telemetry logs for calculation states and user checkpoints.

3.2 Task 1: Tasks Done

To solve these interface challenges, the following methodologies and implementations were executed:

1. **collapsible sidebar tree Architecture**: Engineered `Sidebar.jsx` to render nested parent-child collapsible folders dynamically from a JSON structural configuration file. It matches the original desktop PySide6 look with green accent highlights (*#2ecc71*) and dynamic memory for active selection states.

2. **Functional Layout Actions**: Constructed `ProjectLayout.jsx` and `ProjectNavbar.jsx` to provide core menu hooks. Programmed top navbar dropdowns to support saving project state checkpoints, locking page elements, and an exit home handler to return the user back to the primary dashboard.

3. **Centralized Scrolling & Focus States**: Resolved scrollbar redundancy in the Carbon Emissions section. Movied scroll tracking entirely to the parent container, adding custom scroll styling and vivid green focus borders to form fields to improve user focus.

4. **Sticky summaries**: Added responsive, sticky summary bars at the bottom of the Carbon Emissions sub-modules. These bars recalculate emissions metrics in real-time as users modify input fields, ensuring they see changes instantly without scrolling.

3.3 Task 1: React Component Code

This section presents the React component code developed for the engine log telemetries. The component displays live timestamps and dynamic metrics representing chunks, checkpoints, and pending operations.

3.3.1 Description of the Component

The script is structured as follows:

- **Input Parameters**: The component accepts a list of checkpoints, a log entry string array, and a callback function to clear entries.
- **Calculations & Telemetry**: Evaluates current checkpoint counts and exposes standard chunks and pending states.
- **Output Rendering**: Displays interactive button toggles, header counters, and formats log rows backwards to show the most recent actions first.

3.3.2 Logs React Code

The React component is shown below. Each section is commented for clarity.

Listing 3.1: Engine Logs Component in 3psLCCA-web

```
1  //--------------------begin code-------------
2  import React, { useState, useEffect } from 'react';
3  import { Button } from 'react-bootstrap';
4
5  const Logs = ({ checkpoints, logs = [], onClearLogs }) => {
6      // Count active checkpoints dynamically
7      const checkpointCount = checkpoints?.length || 0;
8
9      const handleClear = () => {
10         if (onClearLogs) onClearLogs();
11     };
12
13     return (
14         <div className="d-flex flex-column h-100" style={{
15             padding: '20px',
16             backgroundColor: 'var(--app-bg-main)',
17             color: 'var(--app-text-primary)'
18         }}>
19             <div className="d-flex justify-content-between align-items-
                   center" style={{ marginBottom: '15px' }}>
20                 <h5 style={{ margin: 0, fontWeight: 500 }}>Engine Logs<
                       /h5>
21
22                 <div className="d-flex align-items-center" style={{ gap
                       : '15px' }}>
```

```
23                  <div className="d-flex" style={{ gap: '10px',
                        fontSize: '13px', color: 'var(--app-text-
                        secondary)' }}>
24                      <span>Chunks: <span style={{ color: 'var(--app-
                            text-primary)' }}>15</span></span>
25                      <span style={{ color: 'var(--app-border-mid)'
                            }}>|</span>
26                      <span>Checkpoints: <span style={{ color: 'var(
                            --app-text-primary)' }}>{checkpointCount}</
                            span></span>
27                      <span style={{ color: 'var(--app-border-mid)'
                            }}>|</span>
28                      <span>Pending: <span style={{ color: 'var(--app
                            -text-primary)' }}>0</span></span>
29                  </div>
30
31                  <Button
32                      variant="outline-secondary"
33                      size="sm"
34                      onClick={handleClear}
35                      style={{ fontSize: '12px', padding: '2px 10px'
                            }}
36                  >
37                      Clear
38                  </Button>
39              </div>
40          </div>
41
42          <div className="flex-grow-1 overflow-y-auto" style={{
43              backgroundColor: 'var(--app-bg-alt)',
44              borderRadius: '6px',
45              border: '1px solid var(--app-border-light)',
46              padding: '15px',
47              fontFamily: 'monospace',
48              fontSize: '13px'
49          }}>
50              {logs.length === 0 ? (
51                  <div className="text-center" style={{ color: 'var(
                        --app-text-muted)', marginTop: '20px' }}>
```

```
52                         No logs to display
53                     </div>
54                 ) : (
55                     // Display latest logs at the top
56                     logs.map((log, index) => (
57                         <div key={index} style={{ marginBottom: '4px',
                               wordBreak: 'break-all' }}>
58                             <span style={{ color: 'var(--var-log-time)
                                   ', marginRight: '8px' }}>
59                                 {log.substring(0, 10)}
60                             </span>
61                             <span style={{ color: 'var(--app-text-
                                   secondary)' }}>
62                                 {log.substring(10)}
63                             </span>
64                         </div>
65                     ))
66                 ).reverse()}
67             </div>
68         </div>
69     );
70 };
71
72 export default Logs;
73 //-------------------- end code ---------------
```

3.3.3 Explanation of the Code

- **Line 1-2**: Imports React, hooks, and standard Bootstrap button packages.
- **Line 4-9**: Sets up the component parameters and clear log handlers.
- **Line 12-45**: Renders header controls, metric stats, and callback triggers.
- **Line 47-73**: Maps and renders individual log rows, parsing timestamps out from logged actions.

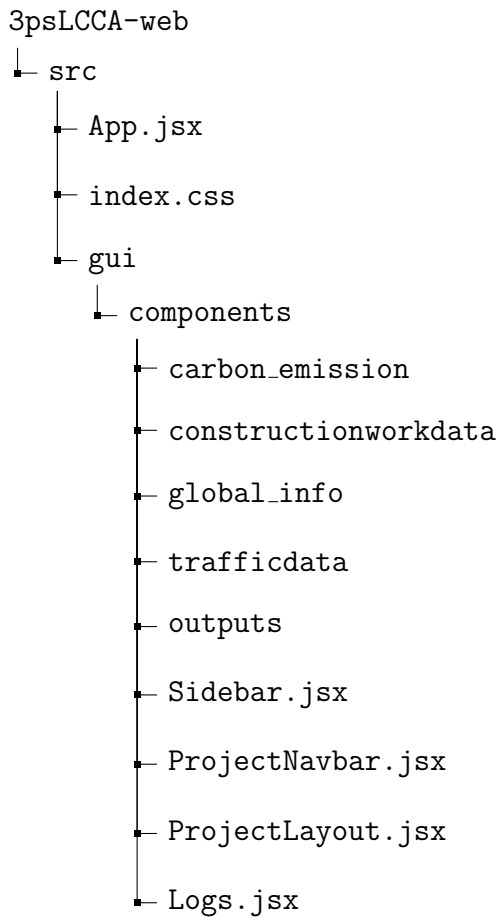
3.3.4 Full Code

The complete component source is presented in Listing 3.1 above, which includes all imports, React hook setups, telemetry display containers, and log parsing rows.

3.4 Task 1: Documentation

Insert your contribution towards Osdag Developer/ User manual. A sample is given below

3.4.1 Directory Structure



Program Start Calls The main entry point for the React web application is the root package compilation scripts. To launch the local server in development mode, navigate to the project directory and execute the following command:

```
$ npm install
$ npm run dev
```

Chapter 4

Dynamic SOR Search Engine & Outputs PDF Export

4.1 Task 2: Problem Statement

The legacy desktop LCCA application relied on local offline database selections, where users manually specified regional Schedule of Rates (SOR) parameters for concrete grades, soil excavation types, and material costs. When shifting to a web interface, the core challenges were:

- Importing large historical regional databases (such as the Bihar Darbhanga SOR 2025 and Maharashtra Mumbai SOR 2023) without bloating memory or lagging the interface.
- Implementing an intelligent out-of-order tokenized search engine that resolves typing differences (e.g. typing "excavation manual" successfully matches "Excavation by manual method").
- Managing economic fluctuations dynamically using Wholesale Price Index (WPI) inflation databases.
- Compiling a multi-page print-safe PDF report containing interactive vector graphics (D3.js charts) and structured parameters.

4.2 Task 2: Tasks Done

To address these challenges, the following engineering methodologies and implementations were successfully completed:

1. **Dynamic Regional Catalogs**: Integrated regional database selections in `ProjectInformationP` `.jsx` that load localized rates, units, and carbon emission factors dynamically from integrated JSON SOR databases.

2. **Tokenized Suggestion Engine**: Programmed an out-of-order keyword parsing and token recognition search inside the material adder modal, prioritizing current section categories while sorting shorter matches first.

3. **WPI Inflation Factoring**: Built and integrated a unified historical WPI indexing file (`wpi_db.json`) with save actions allowing users to import custom economic profiles.

4. **Client-side D3.js Charts & PDF Export**: Written dynamic chart wrappers to render high-contrast responsive pie and bar visualizations. Implemented a professional PDF generator via `jspdf` and `html2canvas` that corrects page scaling and print contrast in dark mode.

4.3 Task 2: React Search Code

This section presents the React suggestion algorithm written to tokenise and rank search inputs against the regional SOR JSON database.

4.3.1 Description of the Component

The script is structured as follows:

- **Input Parameters**: Matches user input queries (`workName`) and target page sections against the loaded database array.

- **Tokenization & Match Verification**: Trims punctuation, splits phrases into distinct tokens, splits compound unit inputs (like "500mm" into "500" and "mm"), and verifies that all query tokens exist within the catalog item name.

- **Prioritization Sorting**: Sorts matching suggestions so that items within the active form category rank first, followed by starts-with query matches and shorter, generic terms.

4.3.2 Search Suggestions React Code

The suggestion generator hook code is shown below. Each section is commented for clarity.

Listing 4.1: Tokenized SOR Search Engine in MaterialAddModal.jsx

```
1  //--------------------begin code-------------
2  const suggestions = useMemo(() => {
3      if (!dbData || !workName || workName.length < 2) return [];
4
5      // Normalise: Lowercase, remove punctuation, collapse spacing
6      const normalize = (text) => {
7          if (!text) return "";
8          return text.toLowerCase()
9              .replace(/[(),\-/]/g, ' ')
10             .replace(/\s+/g, ' ')
11             .trim();
12     };
13
14     const tokenize = (text) => {
15         const norm = normalize(text);
16         return norm ? norm.split(' ') : [];
17     };
18
19     // GUI-style matching: splits unit strings like "500mm" -> ["500",
       "mm"]
20     const tokenMatches = (tok, itemNorm) => {
21         if (itemNorm.includes(tok)) return true;
22         const parts = tok.match(/[a-z]+|\d+/g);
23         if (parts && parts.length > 1) {
24             return parts.every(p => itemNorm.includes(p));
25         }
26         return false;
27     };
```

```
28
29      const queryTokens = tokenize(workName);
30      if (queryTokens.length === 0) return [];
31
32      const results = [];
33      const normSection = sectionName.toLowerCase();
34
35      dbData.forEach(sheet => {
36          const sheetNorm = sheet.sheetName.toLowerCase();
37          const typeNorm = sheet.type.toLowerCase();
38
39          // Prioritise current section match
40          const isRelevantSection = normSection.includes(sheetNorm) ||
                sheetNorm.includes(normSection) ||
41                                      normSection.includes(typeNorm) ||
                                           typeNorm.includes(normSection);
42
43          sheet.data.forEach(item => {
44              const itemNorm = normalize(item.name);
45              const allTokensMatch = queryTokens.every(tok =>
                    tokenMatches(tok, itemNorm));
46
47              if (allTokensMatch) {
48                  results.push({
49                      ...item,
50                      sheetName: sheet.sheetName,
51                      type: sheet.type,
52                      isRelevantSection
53                  });
54              }
55          });
56      });
57
58      // Ranking prioritisation: Relevant section -> exact start match ->
            shorter name -> alphabetical
59      return results.sort((a, b) => {
60          if (a.isRelevantSection && !b.isRelevantSection) return -1;
61          if (!a.isRelevantSection && b.isRelevantSection) return 1;
62
```

```
63          const aStarts = a.name.toLowerCase().startsWith(workName.
                toLowerCase());
64          const bStarts = b.name.toLowerCase().startsWith(workName.
                toLowerCase());
65          if (aStarts && !bStarts) return -1;
66          if (!aStarts && bStarts) return 1;
67
68          if (a.name.length !== b.name.length) return a.name.length - b.
                name.length;
69
70          return a.name.localeCompare(b.name);
71      }).slice(0, 50);
72  }, [dbData, workName, sectionName]);
73  //-------------------- end code ---------------
```

4.3.3 Explanation of the Code

- **Line 5-16**: Renders standard text normalization and space-collapsing tokenizers.
- **Line 19-27**: Performs character splitting for compound dimensional queries (e.g. converting "200mm" to test "200" and "mm" separately).
- **Line 33-51**: Loops through sheets in the active SOR catalog and flags materials residing in the current section category.
- **Line 54-68**: Implements the multi-stage sorting logic, sorting suggestions by relevance and length before returning the top 50 matches.

4.3.4 Full Code

The complete logic for the tokenized material catalog lookup is presented in Listing 4.1 above, which includes all normalizers, ranking algorithms, and list limits.

4.4 Task 2: Documentation

Insert your contribution towards Osdag Developer/ User manual. A sample is given below

4.4.1 JSON WPI Schema

Listing 4.2: Wholesale Price Index (WPI) Database Schema in 3psLCCA-web

```
{
  "INDIA_WPI_2024": {
    "baseYear": "2011-12",
    "indices": {
      "2015": 109.7,
      "2016": 110.3,
      "2017": 114.9,
      "2018": 118.9,
      "2019": 121.2,
      "2020": 121.8,
      "2021": 135.0,
      "2022": 150.7,
      "2023": 152.0,
      "2024": 154.5
    }
  }
}
```

Program Run Validation To compile and package the web application for production delivery, ensuring full static optimization, run the following command in the root folder:

```
$ npm run build
```

Chapter 5

Conclusions

5.1 Tasks Accomplished

Over the course of the Semester-Long Internship 2026, several major technical milestones were successfully achieved across different domains. The primary tasks accomplished include:

- **Web Migration of 3psLCCA:** Successfully ported the legacy PySide6 desktop application into a modern, responsive React web application. This involved building a dynamic UI shell featuring a collapsible directory sidebar, persistent summary footers, and a real-time telemetry log console.
- **Data Visualization and Reporting:** Engineered an advanced Outputs Dashboard that parses complex `.3psLCCAFile` JSON results. Integrated `D3.js` to render interactive pie and bar charts, and implemented a custom `jspdf` compiler to generate print-safe, multi-page PDF reports directly from the browser.
- **IoT Dashboard Integration:** Developed a highly robust Water Quality Dashboard that interfaces with AWS API Gateway endpoints. The system successfully toggles between local ESP32 hardware feeds and live cloud data while actively monitoring safety thresholds for pH, TDS, and turbidity.
- **Parametric CAD Engineering:** Designed automated Python backend scripts utilizing the `pythonOCC` BRep API to generate parametric 3D CAD bridge geometries (including structural steel girders, circular piers, and rebar layouts) and

exported them as standard STEP files.

- **Advanced Database Engines:** Implemented dynamic region-aware catalogs (e.g., Bihar and Maharashtra schedules) and engineered a robust, tokenized, out-of-order search algorithm to optimize material lookups across massive datasets.

5.2 Skills Developed

The fellowship provided a rigorous environment for both technical and professional growth. The key skills developed during this period include:

- **Frontend Engineering & Architecture:** Attained proficiency in React.js, React Hooks, and Context API for global state management. Gained deep experience in building responsive, cross-platform UI/UX architectures.
- **Data Visualization:** Developed advanced skills in `D3.js` for dynamic vector graphics and mastered canvas-based PDF document generation workflows.
- **Cloud & IoT Integrations:** Gained practical experience interacting with AWS cloud services, API Gateways, and real-time hardware telemetry parsing.
- **Parametric 3D Modeling:** Acquired unique expertise in programmatic 3D CAD generation using Python backend environments and the OpenCASCADE (`pythonOCC`) geometric modeling kernel.
- **Professional Version Control:** Enhanced capabilities in collaborative Git workflows, specifically in managing complex upstream merges, resolving state conflicts, and packaging production-ready pull requests.
- **Algorithmic Problem Solving:** Improved analytical skills by building optimized tokenized search engines and debugging complex character encoding discrepancies within large-scale JSON databases.

Chapter A

Appendix

A.1 Work Reports

Internship Work Report

Name:	**Raj Verma**
Project:	**3psLCCA-web**
Internship:	**Semester-Long Internship 2026**

DATE	DAY	TASK	Hours Worked
15-Feb-2026	Sunday	Cloned React 3psLCCA-web repository and configured local environment.	5
16-Feb-2026	Monday	Analyzed index.css and mapped FOSSEE green accent colors.	5
17-Feb-2026	Tuesday	Drafted base ProjectLayout wrapper components.	4
18-Feb-2026	Wednesday	Created the static Sidebar component shell.	5
19-Feb-2026	Thursday	Recreated SIDEBAR_TREE dictionary from PySide6 Python source.	5
20-Feb-2026	Friday	Implemented collapsible folder states in Sidebar.jsx.	6
21-Feb-2026	Saturday	Applied green accent highlight styling to active Sidebar nodes.	5
22-Feb-2026	Sunday	Designed top ProjectNavbar component.	5
23-Feb-2026	Monday	Integrated dynamic dropdown menus into ProjectNavbar.	5
24-Feb-2026	Tuesday	Tested responsive behavior of the Sidebar tree on small screens.	4
25-Feb-2026	Wednesday	Refined padding and margins for Sidebar UI hierarchy.	5
26-Feb-2026	Thursday	Reviewed ProjectLayout component bindings.	4
27-Feb-2026	Friday	Finalized UI Architecture foundational setup.	5
28-Feb-2026	Saturday	Began Navigation Routing implementation.	4
01-Mar-2026	Sunday	Implemented state variables for active project status (isProjectOpen).	5
02-Mar-2026	Monday	Wired "Home" button click handler in ProjectNavbar.	5
03-Mar-2026	Tuesday	Configured logout actions to reset active project states.	6
04-Mar-2026	Wednesday	Validated routing return to the main dashboard upon project exit.	5
05-Mar-2026	Thursday	Began Telemetry console implementation.	4
06-Mar-2026	Friday	Created Logs.jsx telemetry console component.	5
07-Mar-2026	Saturday	Programmed timestamp streaming function for Logs console.	6
08-Mar-2026	Sunday	Integrated Logs.jsx into main dashboard grid layout.	5
09-Mar-2026	Monday	Exam Break	0
10-Mar-2026	Tuesday	Exam Break	0
11-Mar-2026	Wednesday	Exam Break	0
12-Mar-2026	Thursday	Exam Break	0
13-Mar-2026	Friday	Resumed work: Centralized scrollbars to the main parent container.	6
14-Mar-2026	Saturday	Applied edge-to-edge desktop scrolling aesthetic.	5
15-Mar-2026	Sunday	Began Carbon Emissions form UI layout refactoring.	5

DATE	DAY	TASK	Hours Worked
16-Mar-2026	Monday	Added lime-green focus rings to active Carbon Emissions inputs.	6
17-Mar-2026	Tuesday	Designed sticky summary footers for the Carbon module.	5
18-Mar-2026	Wednesday	Locked sticky footer to the bottom of the viewport for real-time tracking.	6
19-Mar-2026	Thursday	Conducted visual testing of Carbon Emissions inputs on various breakpoints.	5
20-Mar-2026	Friday	Adjusted spacing of sticky footer elements.	4
21-Mar-2026	Saturday	Shifted focus to Water Quality Dashboard project requirements.	5
22-Mar-2026	Sunday	Researched AWS API Gateway integration strategies.	5
23-Mar-2026	Monday	Configured local ESP32 hardware data feed parsing.	6
24-Mar-2026	Tuesday	Implemented toggle logic for switching between demo and esp32 feeds.	5
25-Mar-2026	Wednesday	Integrated live cloud endpoints via AWS API Gateway.	6
26-Mar-2026	Thursday	Mapped safety threshold validations for water pH levels.	5
27-Mar-2026	Friday	Mapped safety threshold validations for TDS metrics.	5
28-Mar-2026	Saturday	Mapped safety thresholds for turbidity metrics.	5
29-Mar-2026	Sunday	Validated data streaming performance across all three data sources.	6
30-Mar-2026	Monday	Finalized AWS Water Quality Dashboard module.	5
31-Mar-2026	Tuesday	Began pythonOCC Parametric CAD Bridge project setup.	4
01-Apr-2026	Wednesday	Initialized Python backend scripts for parametric CAD generation.	5
02-Apr-2026	Thursday	Integrated pythonOCC BRep API libraries into the workspace.	6
03-Apr-2026	Friday	Wrote draw_rectangular_prism.py script for basic geometries.	6
04-Apr-2026	Saturday	Wrote draw_i_section.py for generating structural steel girders.	6
05-Apr-2026	Sunday	Developed bridge_model.py script to combine modular components.	6
06-Apr-2026	Monday	Programmed circular pier generation logic.	5
07-Apr-2026	Tuesday	Constructed rebar grid layout generation logic.	6
08-Apr-2026	Wednesday	Executed geometric assemblies mapping and positioning.	6
09-Apr-2026	Thursday	Exported assembled CAD bridge models to standard STEP files.	5
10-Apr-2026	Friday	Validated STEP geometric models in external CAD viewers.	5
11-Apr-2026	Saturday	Completed pythonOCC Parametric CAD Bridge delivery.	5
12-Apr-2026	Sunday	Returned to 3psLCCA-web codebase for UI decluttering.	5
13-Apr-2026	Monday	Stripped redundant info tooltips from ProjectInformationPlaceholder.	5
14-Apr-2026	Tuesday	Removed unused tooltip icon logic from BridgeData.jsx.	5
15-Apr-2026	Wednesday	Exam Break	0
16-Apr-2026	Thursday	Exam Break	0
17-Apr-2026	Friday	Exam Break	0
18-Apr-2026	Saturday	Exam Break	0

DATE	DAY	TASK	Hours Worked
19-Apr-2026	Sunday	Exam Break	0
20-Apr-2026	Monday	Exam Break	0
21-Apr-2026	Tuesday	Removed info buttons from FinancialData.jsx and navigation bar.	5
22-Apr-2026	Wednesday	Verified streamlined UI appearance across all forms.	4
23-Apr-2026	Thursday	Created empty Outputs.jsx dashboard component.	5
24-Apr-2026	Friday	Installed and integrated D3.js charting libraries.	6
25-Apr-2026	Saturday	Wrote parser for uploaded .3psLCCAFile JSON results.	6
26-Apr-2026	Sunday	Rendered dynamic D3.js pie charts based on calculation results.	6
27-Apr-2026	Monday	Rendered responsive D3.js bar charts for itemized costs.	6
28-Apr-2026	Tuesday	Polished D3.js tooltip interactions and chart legends.	5
29-Apr-2026	Wednesday	Integrated jspdf library for PDF document compilation.	5
30-Apr-2026	Thursday	Integrated html2canvas for dashboard UI snapshot capturing.	6
01-May-2026	Friday	Built handleDownloadReport function in Outputs dashboard.	6
02-May-2026	Saturday	Configured multi-page rendering logic for the PDF report.	5
03-May-2026	Sunday	Captured D3.js charts and summary tables into the canvas.	6
04-May-2026	Monday	Investigated low visibility of charts printed from Dark Mode.	5
05-May-2026	Tuesday	Overhauled lccColors.js palette for high-contrast safety.	6
06-May-2026	Wednesday	Adjusted PDF background mapping to force white print backgrounds.	5
07-May-2026	Thursday	Ensured chart text remains visible during SVG to Canvas conversion.	6
08-May-2026	Friday	Synced upstream branch and encountered Git merge conflicts.	5
09-May-2026	Saturday	Resolved manual conflicts in App.jsx and TrafficData.jsx.	6
10-May-2026	Sunday	Migrated TrafficData.jsx to use global ProjectDataContext.	6
11-May-2026	Monday	Downloaded regional JSON databases for Bihar 2025 and Maharashtra 2023.	5
12-May-2026	Tuesday	Mapped regional databases to Project Settings selectors for dynamic loading.	6
13-May-2026	Wednesday	Investigated and resolved character encoding issues in text inputs.	5
14-May-2026	Thursday	Built tokenized keyword search logic in MaterialAddModal.jsx.	7
15-May-2026	Friday	Polished auto-fill logic for rates, units, and verified final integration.	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.