FOSSEE Winter Internship Report

On

Design and Development of OsdagBridge Desktop GUI, CAD Visualization System, and Web Architecture Framework

Submitted by

Yugh Juneja

3rd Year B.Tech Student, Department of CSE

VIT BHOPAL UNIVERSITY

Madhya Pradesh

Under the Guidance of

Prof. Siddhartha Ghosh

Department of Civil Engineering

Indian Institute of Technology Bombay

Mentors:

Nidhi Khare

Aditya Donde

Parth Karia

June 11, 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.

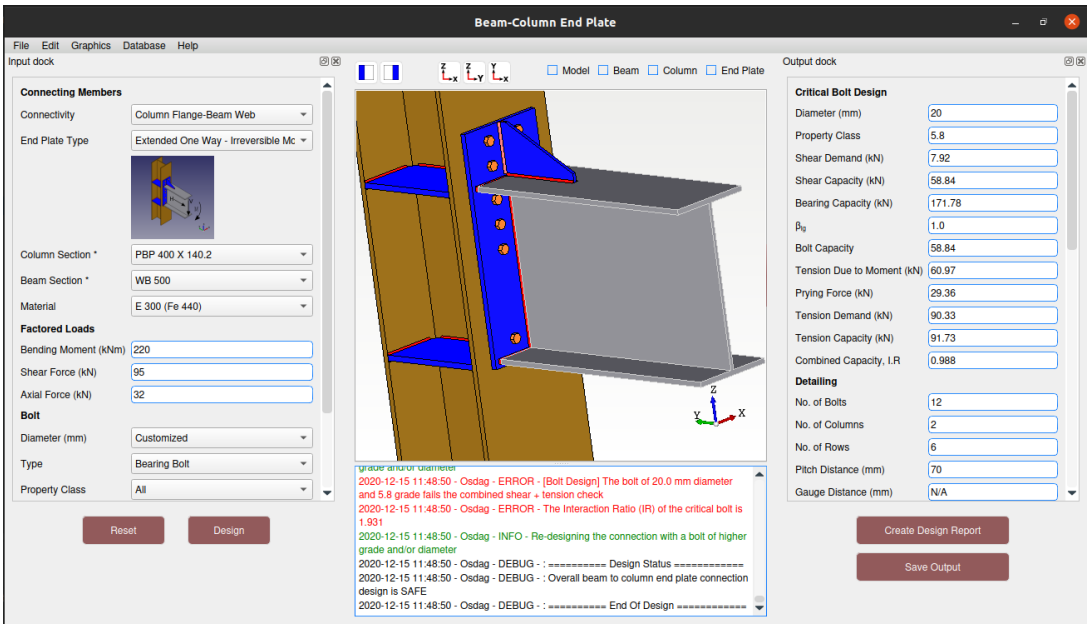

Figure 1.2: Osdag GUI

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

Internship Task 1: Development of OsdagBridge Desktop Application and Structured Implementation of OsdagBridge Web UI

2.1 Task 1: Problem Statement

OsdagBridge required the development of an interactive desktop application capable of visualizing bridge components, managing user inputs, displaying analysis outputs, and providing real-time CAD-based previews. Existing workflows lacked an integrated graphical environment for bridge modelling and visualization.

The objective of this task was to design and develop key desktop GUI modules for OsdagBridge, including CAD visualization interfaces, input and output docks, log management systems, dynamic parameter synchronization, and interactive bridge component views. Additionally, a structured High-Level Design (HLD) and Low-Level Design (LLD) architecture was prepared for the migration of OsdagBridge to a web-based platform using modern frontend and backend technologies. The web architecture aimed to preserve existing engineering logic while enabling scalable, browser-based bridge design workflows.

2.2 Task 1: Tasks Done

The following tasks were performed during this internship activity:

- Developed the OsdagBridge Desktop GUI using PySide6.
- Designed and implemented interactive CAD Cross-Section and Top-View visualization modules for bridge components.
- Developed the Dual CAD View system supporting simultaneous visualization of cross-sectional and top-view bridge geometry.
- Implemented dynamic synchronization between user input fields and CAD visualizations for real-time updates.
- Developed Additional Inputs interfaces with live CAD preview functionality.
- Designed and integrated Log Dock and Output Dock modules for displaying analysis information and user feedback.
- Implemented bridge component rendering including girders, cross-bracing systems, crash barriers, diaphragms, footpaths, and deck geometry.
- Enhanced user interaction through zoom controls, hover detection, view toggles, and dynamic UI updates.
- Designed the High-Level Design (HLD) and Low-Level Design (LLD) architecture for the OsdagBridge Web platform.
- Defined frontend, backend, database, API, CAD rendering, and deployment architecture for the web-based system using React.js, Django REST Framework, PostgreSQL, Redis, Celery, and Konva.

2.3 Task 1: Desktop GUI Development

The desktop application was developed using PySide6 and follows a modular architecture consisting of Input Dock, Output Dock, CAD Visualization Area, Log Window, and Additional Input Dialogs. The system provides real-time interaction between bridge

parameters and CAD representations, allowing users to visualize bridge geometry dynamically as inputs are modified.

The implementation focused on improving usability, modularity, and maintainability while supporting future integration with bridge design and analysis modules.

2.4 Task 1: CAD Visualization Development

A major contribution involved the development of bridge CAD visualization components.

The implemented CAD system includes:

- Cross-Section CAD View
- Top-View CAD Representation
- Dual CAD Workspace
- Independent Zoom Controls
- Dynamic Dimension Rendering
- Hover Detection Zones
- Real-Time Geometry Updates

The visualization modules support rendering of bridge deck geometry, girders, diaphragms, cross-bracing systems, barriers, railings, and footpaths while maintaining synchronization with engineering input parameters.

2.5 Task 1: OsdagBridge Web Architecture

In addition to desktop development, a complete executable HLD and LLD document was prepared for the web version of OsdagBridge.

The proposed architecture consists of:

- React.js with TypeScript for frontend development.
- Django REST Framework for backend services.

- PostgreSQL for project and design data storage.
- Redis and Celery for asynchronous design execution.
- Konva-based CAD rendering engine for browser visualization.
- Reuse of existing Python engineering logic and standards modules.

The architecture preserves the existing desktop domain model while enabling scalable browser-based bridge design workflows. Major system components include project management services, CAD scene builders, design engines, report generation services, and workspace visualization modules such as BridgeDualCADWidget, CrossSectionCADWidget, and TopViewCADWidget.

Chapter 3

Internship Task 2: Architecture Design and Development Planning for OsdagBridge Web Platform

3.1 Task 2: Problem Statement

OsdagBridge was originally developed as a desktop application with integrated CAD visualization, engineering calculations, and project management capabilities. To improve accessibility, scalability, and platform independence, a web-based version of OsdagBridge was proposed.

The objective of this task was to design the High-Level Design (HLD) and Low-Level Design (LLD) architecture for the OsdagBridge Web platform. The architecture needed to preserve the existing Python-based engineering logic while introducing a modern web-based workspace capable of supporting CAD visualization, project management, bridge design workflows, and future cloud deployment.

3.2 Task 2: Tasks Done

The following tasks were carried out during this activity:

- Designed the High-Level Design (HLD) architecture for OsdagBridge Web.
- Prepared the Low-Level Design (LLD) for frontend, backend, database, and CAD

subsystems.

- Defined the complete technology stack using React.js, TypeScript, Django REST Framework, PostgreSQL, Redis, Celery, and Konva.
- Designed reusable frontend component architecture for CAD visualization and workspace management.
- Defined API contracts for project management, CAD scene generation, design execution, and report generation.
- Planned migration strategies to preserve existing desktop engineering logic and naming conventions.
- Designed scene-based CAD rendering architecture using JSON-driven visualization.
- Defined project versioning, file management, logging, and report generation workflows.
- Structured frontend state management and backend service boundaries for scalable development.

3.2.1 High-Level System Architecture

The proposed architecture follows a layered design consisting of:

- React.js and TypeScript frontend.
- Konva-based 2D CAD rendering engine.
- Django REST API services.
- Reusable Python engineering engine.
- PostgreSQL database.
- Redis and Celery for asynchronous processing.

This architecture allows separation of user interface components, CAD rendering services, engineering calculations, and persistence layers while maintaining scalability and maintainability.

3.2.2 Frontend Workspace Design

The frontend architecture was designed around a workspace concept similar to the desktop application.

Major frontend modules include:

- BridgeDualCADWidget
- CrossSectionCADWidget
- TopViewCADWidget
- Input Dock
- Output Dock
- Log Panel
- Additional Inputs Dialog

The workspace supports live parameter updates, CAD synchronization, independent view controls, hover detection, zoom functionality, and project management features.

3.2.3 CAD Rendering Architecture

A scene-driven CAD rendering strategy was proposed for the web platform.

Instead of embedding engineering logic inside rendering components, backend services generate Scene JSON objects containing:

- Geometry definitions
- Dimensions
- Labels
- Hover metadata
- Visibility states

The frontend renderer consumes the generated Scene JSON and visualizes bridge components without performing engineering calculations. This approach improves modularity and simplifies future maintenance.

3.2.4 Backend Service Design

The backend architecture was divided into multiple service modules:

- Project Management Service
- CAD Scene Builder Service
- Design Execution Service
- Report Generation Service
- File Management Service

The backend preserves the existing Python engineering engine and exposes its functionality through REST APIs. Long-running design operations are handled asynchronously using Celery and Redis.

3.2.5 API Design and Data Flow

The web platform uses REST APIs to communicate between frontend and backend modules.

The workflow consists of:

- User updates project parameters.
- Frontend sends debounced API requests.
- Backend validates and stores project data.
- CAD scene builders generate updated Scene JSON.
- Frontend re-renders CAD views.
- Design runs are executed asynchronously through job queues.
- Results are displayed in the Output Dock and Log Window.

This architecture enables real-time interaction while maintaining system scalability.

3.2.6 Documentation and Design Deliverables

Comprehensive HLD and LLD documents were prepared covering:

- System Context Diagram
- Module Boundaries
- Repository Structure
- Frontend Components
- Backend Services
- Database Design
- Scene JSON Contracts
- API Specifications
- Execution Plan
- Risk Assessment and Mitigation Strategies

These documents serve as the foundation for future development of the OsdagBridge Web platform.

3.3 Task 2: Documentation

3.3.1 Module Workflow Summary

- User accesses the web-based workspace.
- Project parameters are managed through Input Dock interfaces.
- Backend services generate Scene JSON for CAD rendering.
- Cross-Section and Top-View renderers visualize bridge geometry.
- Design jobs execute through reusable Python engineering modules.
- Results are displayed through Output Dock and Log Window components.
- Reports and project files are generated and stored through backend services.

Chapter 4

Internship Task 3: Development of Output Dock Interface and Advanced Visualization Components in OsdagBridge Desktop

4.1 Task 3: Problem Statement

The OsdagBridge Desktop application required a comprehensive output and visualization system to present analysis results, design information, and bridge component data in an organized and user-friendly manner. Existing interfaces lacked a structured mechanism for displaying member forces, load combinations, design results, and visualization controls.

The objective of this task was to design and develop an advanced Output Dock interface along with supporting visualization components that enable users to interact with analysis results, control display options, and efficiently navigate bridge design information within the OsdagBridge Desktop environment.

4.2 Task 3: Tasks Done

The following tasks were carried out during this activity:

- Designed and developed the Output Dock user interface using PySide6.
- Implemented Analysis Results panels for displaying bridge analysis information.
- Developed member selection and load combination controls.
- Created force visualization controls for displaying shear forces, moments, torsion, and displacement results.
- Implemented display option controls for maximum, minimum, and controlling utilization ratio views.
- Enhanced UI consistency through reusable styling and component-based design.
- Integrated Output Dock functionality with the OsdagBridge Desktop workspace.
- Improved interaction between CAD visualization components and analysis result displays.
- Performed UI testing and validation to ensure usability and maintainability.

4.2.1 Analysis Results Interface

A dedicated Analysis Results section was developed to provide structured access to bridge analysis outputs. The interface allows users to select bridge members and load combinations while dynamically updating displayed results.

The implementation supports:

- Member selection controls.
- Load combination selection.
- Analysis result filtering.
- Interactive force visualization.

4.2.2 Force Visualization Controls

A specialized force selection interface was developed to allow users to visualize various structural response parameters.

The supported parameters include:

- Shear Forces (Vx, Vy, Vz)
- Bending Moments (My, Mz)
- Torsional Effects (Tx)
- Displacements (Dx, Dy, Dz)

Radio-button based controls were implemented to ensure intuitive selection and result visualization.

4.2.3 Display Options Development

A dedicated display management section was implemented to provide flexibility in result interpretation.

The interface includes:

- Maximum Result Display
- Minimum Result Display
- Controlling Utilization Ratio Display

These controls help users quickly identify governing design conditions and critical structural responses.

4.2.4 Workspace Integration

The Output Dock was integrated into the overall OsdagBridge workspace architecture and connected with:

- Input Dock
- CAD Visualization Area
- Log Window
- Additional Inputs Dialog

This integration ensures consistent communication between user inputs, CAD visualizations, and displayed analysis results.

4.2.5 User Interface Enhancements

Several improvements were made to enhance user experience:

- Consistent styling across all dock components.
- Improved layout organization.
- Dynamic widget generation.
- Better readability of analysis information.
- Enhanced responsiveness of interface components.

4.2.6 Testing and Validation

The developed Output Dock system was tested extensively to verify:

- Correct rendering of analysis result controls.
- Proper interaction between UI components.
- Stable integration with CAD visualization modules.
- Consistent display behavior across different bridge configurations.
- Overall usability and performance of the interface.

4.3 Task 3: Documentation

4.3.1 Directory Structure

```
src/
|-- osdagbridge/
|   |-- desktop/
|       |-- ui/
|           |-- docks/
|               |-- output_dock.py
|               |-- input_dock.py
```

```
|                |-- log_dock.py
|            |-- dialogs/
|            |-- template_page.py
```

4.3.2 Module Workflow Summary

- User performs bridge modelling through the Input Dock.
- CAD views update dynamically based on input parameters.
- Analysis results are displayed through the Output Dock.
- Users select members and load combinations for detailed inspection.
- Force visualization controls display structural response quantities.
- Display options help identify governing design conditions.
- Results remain synchronized with CAD and workspace components.

Chapter 5

Internship Task 4: Development of CAD Visualization Framework and Dynamic Parameter Synchronization for OsdagBridge

5.1 Task 4: Problem Statement

Bridge design applications require accurate graphical visualization of structural components to improve user understanding and facilitate design verification. Prior to this work, OsdagBridge required a comprehensive CAD visualization framework capable of supporting interactive cross-sectional views, top-view representations, parameter synchronization, and future migration to web-based rendering systems.

The objective of this task was to develop a scalable CAD visualization architecture for OsdagBridge Desktop while establishing reusable concepts for future web-based CAD rendering. The work focused on dynamic geometry generation, parameter synchronization, hover interactions, dimension rendering, dual-view visualization, and scene-driven CAD workflows.

5.2 Task 4: Tasks Done

The following tasks were performed during this activity:

- Developed Cross-Section CAD visualization modules for bridge geometry.
- Developed Top-View CAD rendering components.
- Implemented Dual CAD View architecture supporting simultaneous visualization of multiple bridge views.
- Designed dynamic parameter synchronization between user inputs and CAD geometry.
- Implemented dimension annotation and measurement visualization.
- Developed hover interaction zones for bridge component inspection.
- Integrated CAD visualization with Input Dock and Additional Inputs interfaces.
- Standardized color schemes and graphical representation of bridge components.
- Designed scene-driven rendering concepts later adopted in the OsdagBridge Web architecture.
- Contributed to CAD architecture planning documented in the OsdagBridge HLD and LLD reports.

5.2.1 Cross-Section CAD Development

A dedicated CAD module was developed to render bridge cross-sections dynamically based on engineering parameters supplied by the user.

The visualization system supports:

- Bridge deck representation.
- Steel girder visualization.
- Cross-bracing systems.
- End diaphragms.
- Crash barriers.
- Footpaths and railings.

- Bearings and support elements.

The generated geometry updates automatically whenever design parameters are modified.

5.2.2 Top-View CAD Development

A complementary top-view visualization system was implemented to provide a plan representation of the bridge structure.

The module supports:

- Girder layout visualization.
- Span representation.
- Cross-bracing positioning.
- Deck geometry display.
- Interactive component highlighting.

The top-view renderer shares common visualization rules with the cross-sectional CAD system, ensuring consistency throughout the application.

5.2.3 Dual CAD Workspace

A dual-view CAD workspace was developed to allow simultaneous display of:

- Cross-Section View
- Top View

Independent zoom controls, view toggles, and visibility management features were implemented to improve user interaction and navigation within large bridge models.

5.2.4 Dynamic Parameter Synchronization

A real-time synchronization framework was developed between application inputs and CAD visualization components.

Whenever bridge parameters are modified:

- Input values are validated.
- Updated geometry parameters are generated.
- CAD widgets receive revised configuration data.
- Visual representations are automatically refreshed.

This approach significantly improves usability and provides immediate graphical feedback to users.

5.2.5 Interactive Visualization Features

Several interactive features were implemented to improve the visualization experience:

- Hover Detection Zones
- Component Highlighting
- Dimension Display
- Dynamic Scaling
- View Toggle Controls
- Real-Time Geometry Updates

These features allow users to inspect bridge components more effectively and understand geometric relationships within the structure.

5.2.6 CAD Architecture Planning for Web Platform

The concepts developed during desktop CAD implementation were later extended into the web architecture planning process.

A scene-driven rendering model was proposed in which:

- Backend services generate Scene JSON.
- Geometry definitions are stored separately from rendering logic.
- CAD renderers act as visualization engines only.
- Engineering calculations remain within reusable Python modules.

This architecture forms the basis for future CAD implementation in the OsdagBridge Web platform using React.js and Konva.

5.2.7 Testing and Validation

The visualization framework was tested using multiple bridge configurations to verify:

- Correct geometry generation.
- Proper synchronization of user inputs.
- Accurate rendering of bridge components.
- Consistent scaling behavior.
- Stable hover interactions.
- Reliable dual-view operation.

5.3 Task 4: Documentation

5.3.1 Directory Structure

```
src/
|-- osdagbridge/
```

```
|   |-- desktop/
|       |-- ui/
|           |-- docks/
|               |-- cad_cross_section.py
|               |-- cad_top_view.py
|               |-- cad_dual_view.py
|               |-- input_dock.py
|           |-- dialogs/
|               |-- additional_inputs.py
|           |-- template_page.py
```

5.3.2 Visualization Workflow Summary

- User enters bridge parameters through Input Dock.
- Parameters are validated and synchronized with CAD modules.
- Cross-Section and Top-View geometries are generated dynamically.
- Hover zones and dimensions are updated automatically.
- Dual CAD Workspace displays synchronized visualizations.
- Scene generation concepts support future migration to the OsdagBridge Web platform.

Chapter 6

Internship Task 5: System Integration, State Management, and Workflow Synchronization in OsdagBridge

6.1 Task 5: Problem Statement

The OsdagBridge platform consists of multiple interconnected modules including Input Dock, Additional Inputs Dialog, CAD Visualization Components, Output Dock, Log Window, and Design Execution Services. As the application evolved, there was a need to establish a unified workflow that could synchronize user inputs, visualization components, analysis outputs, and backend services efficiently.

The objective of this task was to design and implement an integrated workflow architecture that enables seamless communication between various desktop modules while also establishing design principles for future web-based implementation. The focus was on state synchronization, data flow management, workspace integration, and maintainable software architecture.

6.2 Task 5: Tasks Done

The following activities were performed during this task:

- Integrated Input Dock, CAD Workspace, Output Dock, and Log Window into a

unified workflow.

- Developed centralized state synchronization mechanisms for bridge parameters.
- Implemented real-time communication between user inputs and CAD visualization modules.
- Designed workflow management for Additional Inputs and parameter updates.
- Established architecture for project state management and future web migration.
- Developed event-driven update mechanisms for dynamic user interaction.
- Designed component communication patterns between frontend modules.
- Contributed to workflow definitions documented in the OsdagBridge HLD and LLD architecture documents.

6.3 Task 5: Input Management System

A structured input management framework was developed to collect, validate, and distribute bridge parameters throughout the application.

The system handles:

- Span parameters
- Carriageway dimensions
- Girder configuration
- Deck geometry
- Footpath properties
- Cross-bracing parameters
- Additional bridge configuration settings

The collected inputs serve as the central source of information for CAD visualization and future design calculations.

6.4 Task 5: Real-Time Synchronization Framework

A dynamic synchronization mechanism was implemented to ensure immediate propagation of user changes throughout the application.

The workflow follows:

- User modifies bridge parameters.
- Input validation is performed.
- State information is updated.
- CAD visualization receives updated parameters.
- Cross-section and top-view renderers regenerate geometry.
- Output components remain synchronized with the latest project state.

This architecture significantly improves responsiveness and user experience.

6.5 Task 5: Workspace Integration

The desktop workspace was designed as an integrated environment consisting of:

- Input Dock
- Additional Inputs Dialog
- BridgeDualCADWidget
- CrossSectionCADWidget
- TopViewCADWidget
- Output Dock
- Log Window

All components communicate through centralized update mechanisms, ensuring consistency across the application.

6.6 Task 5: State Management Architecture

As part of the OsdagBridge Web planning effort, a formal state management strategy was proposed.

The architecture defines separate state domains for:

- Project State
- CAD State
- Log State
- User Interface State

This separation minimizes coupling between modules and improves maintainability. The same principles are applicable to both desktop and web implementations.

6.7 Task 5: System Workflow Design

A complete system workflow was designed covering:

- Project creation
- Parameter modification
- CAD scene generation
- Visualization updates
- Design execution
- Result presentation
- Report generation

The workflow establishes a clear separation between visualization, engineering logic, persistence, and user interaction layers.

6.8 Task 5: Future Web Integration Planning

The synchronization concepts developed during desktop implementation were incorporated into the web architecture design.

Key planning decisions included:

- Debounced parameter updates.
- Scene JSON based CAD rendering.
- Separation of frontend and backend responsibilities.
- Reuse of existing Python engineering modules.
- REST API based communication.
- Asynchronous design execution using Celery and Redis.

These decisions form the foundation for scalable browser-based bridge design workflows.

6.9 Task 5: Documentation

6.9.1 Integrated Workflow Summary

- User enters bridge parameters through the Input Dock.
- Additional Inputs provide detailed bridge configuration settings.
- Centralized state management stores project information.
- CAD modules generate synchronized visualizations.
- Output Dock displays engineering results.
- Log Window records workflow activity and system messages.
- Future web services consume the same project and scene structures.

6.10 Task 5: Summary

This task established the integration framework connecting all major OsdagBridge modules. The developed synchronization mechanisms, state management strategies, and workflow architecture improve maintainability, enhance user interaction, and provide a strong foundation for future web-based deployment of the platform.

Chapter 7

Conclusions

7.1 Tasks Accomplished

During the course of the semester-long internship, significant contributions were made towards the development of the OsdagBridge platform, focusing on desktop application development, CAD visualization systems, user interface design, system integration, and web architecture planning. The work carried out throughout the internship aimed to improve the usability, scalability, maintainability, and future extensibility of the bridge design software.

One of the primary contributions was the development and enhancement of the OsdagBridge Desktop graphical user interface. Multiple user interface components were designed and integrated, including Input Dock, Output Dock, Log Window, Additional Inputs Dialog, and workspace management features. These components were developed to provide a structured and user-friendly environment for bridge modelling and visualization.

A major portion of the internship involved the development of interactive CAD visualization systems. Dedicated Cross-Section and Top-View CAD modules were implemented to dynamically render bridge geometry based on user-defined parameters. The visualization framework supports real-time updates, dimension rendering, component highlighting, hover interactions, and synchronized display of multiple bridge views. The development of the Dual CAD Workspace further enhanced the user experience by allowing simultaneous visualization of different bridge representations.

Considerable effort was also devoted to implementing dynamic parameter synchro-

nization throughout the application. Mechanisms were developed to ensure that changes made through input forms are automatically propagated to CAD visualization modules and related interface components. This significantly improved responsiveness and workflow efficiency while maintaining consistency across the application.

Another important contribution involved the design and implementation of advanced user interface components for displaying analysis information and visualization controls. The Output Dock was developed to provide organized access to structural analysis results, load combinations, member selection tools, and visualization options. Several improvements were also made to workspace navigation, usability, and component interaction.

In addition to desktop development, extensive work was carried out on the planning and architectural design of the OsdagBridge Web platform. Comprehensive High-Level Design (HLD) and Low-Level Design (LLD) documents were prepared to define the future architecture of the system. The proposed architecture utilizes React.js, TypeScript, Django REST Framework, PostgreSQL, Redis, Celery, and Konva while preserving the existing Python-based engineering logic. Detailed designs were prepared for frontend components, backend services, database structures, CAD rendering workflows, API contracts, and deployment strategies.

The internship also involved integration testing, workflow validation, debugging, and documentation activities. Various desktop modules were tested to verify correct interaction between user inputs, CAD visualization systems, output displays, and workspace components. The resulting improvements contributed to increased software reliability, maintainability, and user experience.

Overall, the work completed during the internship established a strong technical foundation for both the current desktop application and the future web-based evolution of OsdagBridge.

7.2 Skills Developed

The internship provided valuable exposure to large-scale software development, graphical user interface engineering, CAD visualization systems, software architecture design, and collaborative open-source development practices.

Strong proficiency was developed in Python programming through extensive work

on desktop application development and system integration. Experience was gained in designing modular software components, implementing maintainable code structures, debugging complex systems, and integrating multiple software modules within a unified application framework.

Significant practical experience was acquired in desktop GUI development using PySide6. This included designing layouts, implementing event-driven workflows, managing user interactions, creating reusable interface components, and improving application usability. Working on large-scale GUI systems enhanced understanding of software architecture and interface design principles.

The internship also provided substantial exposure to CAD visualization concepts. Through the development of Cross-Section and Top-View rendering systems, a deeper understanding was gained of geometry representation, graphical rendering, coordinate systems, interactive visualization techniques, and engineering software workflows.

Another important learning outcome was software architecture and system design. Preparing High-Level Design and Low-Level Design documentation for the OsdagBridge Web platform strengthened skills in designing scalable software systems, defining API contracts, planning database architectures, organizing frontend-backend communication, and creating maintainable development workflows.

Practical experience was also gained in modern web technologies and system planning, including React.js, TypeScript, Django REST Framework, PostgreSQL, Redis, Celery, REST APIs, and scene-based rendering architectures. Understanding how desktop engineering applications can be transformed into scalable web platforms provided valuable insight into full-stack software engineering practices.

Beyond technical skills, the internship strengthened abilities in problem solving, debugging, documentation, version control, code reviews, and collaborative software development. Regular interaction with mentors and contributors improved communication skills, project management awareness, and understanding of professional software development workflows.

Overall, the internship was a highly rewarding learning experience that significantly enhanced software engineering, system design, visualization, and architecture development skills. The knowledge and experience gained through this internship provide a strong foundation for future work in software engineering, CAD systems, engineering

applications, and large-scale technology development.

Chapter A

Appendix

A.1 Internship Work Reports

This section contains the detailed day-wise work reports maintained throughout the internship period. The reports provide a chronological record of development activities, architectural planning, CAD implementation, user interface development, testing, documentation, and integration work carried out during the development of the OsdagBridge platform.

The work reports include information regarding:

- Desktop GUI development activities.
- CAD visualization framework implementation.
- Output Dock and Log Window development.
- Dynamic parameter synchronization and workflow integration.
- High-Level Design (HLD) and Low-Level Design (LLD) preparation.
- Web platform architecture planning.
- Documentation, testing, and validation activities.

Internship Work Report

Name: Yugh Juneja
Project: OsdagBridge
Internship: FOSSEE Semester Long Internship (2026)

Date	Day	Task	Hours
05-Jan-2026	Monday	Developed and refined OsdagBridge Desktop GUI components.	4
06-Jan-2026	Tuesday	Implemented and tested CAD Cross-Section visualization features.	5
07-Jan-2026	Wednesday	Enhanced Top-View CAD rendering and interaction workflows.	6
08-Jan-2026	Thursday	Integrated dynamic parameter synchronization with CAD modules.	7
09-Jan-2026	Friday	Developed Input Dock and Additional Inputs functionality.	8
10-Jan-2026	Saturday	Implemented Output Dock user interface and result visualization.	4
11-Jan-2026	Sunday	Weekly Off.	0
12-Jan-2026	Monday	Enhanced Log Window and workspace integration features.	5
13-Jan-2026	Tuesday	Designed reusable UI components and improved user workflows.	6
14-Jan-2026	Wednesday	Prepared HLD and LLD documentation for OsdagBridge Web.	7
15-Jan-2026	Thursday	Designed React.js and Django-based web architecture modules.	8
16-Jan-2026	Friday	Defined Scene JSON contracts and CAD rendering workflows.	4
17-Jan-2026	Saturday	Documented API design, database schema, and deployment plans.	5
18-Jan-2026	Sunday	Weekly Off.	0
19-Jan-2026	Monday	Performed integration testing and resolved UI issues.	6
20-Jan-2026	Tuesday	Reviewed code changes, documentation, and mentor feedback.	7
21-Jan-2026	Wednesday	Prepared technical documentation and internship deliverables.	8
22-Jan-2026	Thursday	Developed and refined OsdagBridge Desktop GUI components.	4
23-Jan-2026	Friday	Implemented and tested CAD Cross-Section visualization features.	5
24-Jan-2026	Saturday	Enhanced Top-View CAD rendering and interaction workflows.	6
25-Jan-2026	Sunday	Weekly Off.	0
26-Jan-2026	Monday	Integrated dynamic parameter synchronization with CAD modules.	7
27-Jan-2026	Tuesday	Developed Input Dock and Additional Inputs functionality.	8

28-Jan-2026	Wednesday	Implemented Output Dock user interface and result visualization.	4
29-Jan-2026	Thursday	Enhanced Log Window and workspace integration features.	5
30-Jan-2026	Friday	Designed reusable UI components and improved user workflows.	6
31-Jan-2026	Saturday	Prepared HLD and LLD documentation for OsdagBridge Web.	7
01-Feb-2026	Sunday	Weekly Off.	0
02-Feb-2026	Monday	Designed React.js and Django-based web architecture modules.	8
03-Feb-2026	Tuesday	Defined Scene JSON contracts and CAD rendering workflows.	4
04-Feb-2026	Wednesday	Documented API design, database schema, and deployment plans.	5
05-Feb-2026	Thursday	Performed integration testing and resolved UI issues.	6
06-Feb-2026	Friday	Reviewed code changes, documentation, and mentor feedback.	7
07-Feb-2026	Saturday	Prepared technical documentation and internship deliverables.	8
08-Feb-2026	Sunday	Weekly Off.	0
09-Feb-2026	Monday	Developed and refined OsdagBridge Desktop GUI components.	4
10-Feb-2026	Tuesday	Implemented and tested CAD Cross-Section visualization features.	5
11-Feb-2026	Wednesday	Enhanced Top-View CAD rendering and interaction workflows.	6
12-Feb-2026	Thursday	Integrated dynamic parameter synchronization with CAD modules.	7
13-Feb-2026	Friday	Developed Input Dock and Additional Inputs functionality.	8
14-Feb-2026	Saturday	Implemented Output Dock user interface and result visualization.	4
15-Feb-2026	Sunday	Weekly Off.	0
16-Feb-2026	Monday	Enhanced Log Window and workspace integration features.	5
17-Feb-2026	Tuesday	Designed reusable UI components and improved user workflows.	6
18-Feb-2026	Wednesday	Prepared HLD and LLD documentation for OsdagBridge Web.	7
19-Feb-2026	Thursday	Designed React.js and Django-based web architecture modules.	8
20-Feb-2026	Friday	Defined Scene JSON contracts and CAD rendering workflows.	4
21-Feb-2026	Saturday	Documented API design, database schema, and deployment plans.	5
22-Feb-2026	Sunday	Weekly Off.	0
23-Feb-2026	Monday	Performed integration testing and resolved UI issues.	6

24-Feb-2026	Tuesday	Reviewed code changes, documentation, and mentor feedback.	7
25-Feb-2026	Wednesday	Prepared technical documentation and internship deliverables.	8
26-Feb-2026	Thursday	Developed and refined OsdagBridge Desktop GUI components.	4
27-Feb-2026	Friday	Implemented and tested CAD Cross-Section visualization features.	5
28-Feb-2026	Saturday	Enhanced Top-View CAD rendering and interaction workflows.	6
01-Mar-2026	Sunday	Weekly Off.	0
02-Mar-2026	Monday	Integrated dynamic parameter synchronization with CAD modules.	7
03-Mar-2026	Tuesday	Developed Input Dock and Additional Inputs functionality.	8
04-Mar-2026	Wednesday	Implemented Output Dock user interface and result visualization.	4
05-Mar-2026	Thursday	Enhanced Log Window and workspace integration features.	5
06-Mar-2026	Friday	Designed reusable UI components and improved user workflows.	6
07-Mar-2026	Saturday	Exam Leave.	0
08-Mar-2026	Sunday	Weekly Off.	0
09-Mar-2026	Monday	Exam Leave.	0
10-Mar-2026	Tuesday	Exam Leave.	0
11-Mar-2026	Wednesday	Exam Leave.	0
12-Mar-2026	Thursday	Exam Leave.	0
13-Mar-2026	Friday	Exam Leave.	0
14-Mar-2026	Saturday	Exam Leave.	0
15-Mar-2026	Sunday	Weekly Off.	0
16-Mar-2026	Monday	Exam Leave.	0
17-Mar-2026	Tuesday	Exam Leave.	0
18-Mar-2026	Wednesday	Exam Leave.	0
19-Mar-2026	Thursday	Exam Leave.	0
20-Mar-2026	Friday	Exam Leave.	0
21-Mar-2026	Saturday	Exam Leave.	0
22-Mar-2026	Sunday	Weekly Off.	0
23-Mar-2026	Monday	Prepared HLD and LLD documentation for OsdagBridge Web.	7
24-Mar-2026	Tuesday	Designed React.js and Django-based web architecture modules.	8
25-Mar-2026	Wednesday	Defined Scene JSON contracts and CAD rendering workflows.	4
26-Mar-2026	Thursday	Documented API design, database schema, and deployment plans.	5
27-Mar-2026	Friday	Performed integration testing and resolved UI issues.	6

28-Mar-2026	Saturday	Reviewed code changes, documentation, and mentor feedback.	7
29-Mar-2026	Sunday	Weekly Off.	0
30-Mar-2026	Monday	Prepared technical documentation and internship deliverables.	8
31-Mar-2026	Tuesday	Developed and refined OsdagBridge Desktop GUI components.	4
01-Apr-2026	Wednesday	Implemented and tested CAD Cross-Section visualization features.	5
02-Apr-2026	Thursday	Enhanced Top-View CAD rendering and interaction workflows.	6
03-Apr-2026	Friday	Integrated dynamic parameter synchronization with CAD modules.	7
04-Apr-2026	Saturday	Developed Input Dock and Additional Inputs functionality.	8
05-Apr-2026	Sunday	Weekly Off.	0
06-Apr-2026	Monday	Implemented Output Dock user interface and result visualization.	4
07-Apr-2026	Tuesday	Enhanced Log Window and workspace integration features.	5
08-Apr-2026	Wednesday	Designed reusable UI components and improved user workflows.	6
09-Apr-2026	Thursday	Prepared HLD and LLD documentation for OsdagBridge Web.	7
10-Apr-2026	Friday	Designed React.js and Django-based web architecture modules.	8
11-Apr-2026	Saturday	Defined Scene JSON contracts and CAD rendering workflows.	4
12-Apr-2026	Sunday	Weekly Off.	0
13-Apr-2026	Monday	Documented API design, database schema, and deployment plans.	5
14-Apr-2026	Tuesday	Performed integration testing and resolved UI issues.	6
15-Apr-2026	Wednesday	Reviewed code changes, documentation, and mentor feedback.	7
16-Apr-2026	Thursday	Prepared technical documentation and internship deliverables.	8
17-Apr-2026	Friday	Exam Leave.	0
18-Apr-2026	Saturday	Exam Leave.	0
19-Apr-2026	Sunday	Weekly Off.	0
20-Apr-2026	Monday	Exam Leave.	0
21-Apr-2026	Tuesday	Exam Leave.	0
22-Apr-2026	Wednesday	Exam Leave.	0
23-Apr-2026	Thursday	Exam Leave.	0
24-Apr-2026	Friday	Exam Leave.	0
25-Apr-2026	Saturday	Exam Leave.	0
26-Apr-2026	Sunday	Weekly Off.	0
27-Apr-2026	Monday	Exam Leave.	0

28-Apr-2026	Tuesday	Exam Leave.	0
29-Apr-2026	Wednesday	Exam Leave.	0
30-Apr-2026	Thursday	Exam Leave.	0
01-May-2026	Friday	Developed and refined OsdagBridge Desktop GUI components.	4
02-May-2026	Saturday	Implemented and tested CAD Cross-Section visualization features.	5
03-May-2026	Sunday	Weekly Off.	0
04-May-2026	Monday	Enhanced Top-View CAD rendering and interaction workflows.	6
05-May-2026	Tuesday	Integrated dynamic parameter synchronization with CAD modules.	7
06-May-2026	Wednesday	Developed Input Dock and Additional Inputs functionality.	8
07-May-2026	Thursday	Implemented Output Dock user interface and result visualization.	4
08-May-2026	Friday	Enhanced Log Window and workspace integration features.	5
09-May-2026	Saturday	Designed reusable UI components and improved user workflows.	6
10-May-2026	Sunday	Weekly Off.	0
11-May-2026	Monday	Prepared HLD and LLD documentation for OsdagBridge Web.	7
12-May-2026	Tuesday	Designed React.js and Django-based web architecture modules.	8
13-May-2026	Wednesday	Defined Scene JSON contracts and CAD rendering workflows.	4
14-May-2026	Thursday	Documented API design, database schema, and deployment plans.	5
15-May-2026	Friday	Performed integration testing and resolved UI issues.	6
16-May-2026	Saturday	Reviewed code changes, documentation, and mentor feedback.	7
17-May-2026	Sunday	Weekly Off.	0
18-May-2026	Monday	Prepared technical documentation and internship deliverables.	8
19-May-2026	Tuesday	Developed and refined OsdagBridge Desktop GUI components.	4
20-May-2026	Wednesday	Implemented and tested CAD Cross-Section visualization features.	5

A.2 GitHub Contributions and Pull Requests

Throughout the internship, contributions were made to the OsdagBridge repository through feature development, architectural improvements, CAD visualization enhancements, user interface development, and workflow integration. These contributions were reviewed and merged through multiple pull requests.

A.2.1 Major Pull Requests

- **#9 — Implemented Log Window and CAD Cross Section UI**

  Developed the Cross-Section CAD visualization framework, Dual CAD Workspace components, Log Window integration, and supporting user interface infrastructure. The contribution included implementation of bridge geometry rendering, dynamic visualization updates, and CAD workspace controls.

- **#21 — Implemented Dynamic UI for Additional Inputs and CAD Enhancements** Developed dynamic user interface components for Additional Inputs, implemented live CAD preview functionality, improved cross-section visualization behavior, enhanced top-view rendering, and standardized visualization workflows.

- **#37 — Added Output Dock User Interface** Designed and implemented the Output Dock system for displaying analysis results, member selection controls, load combinations, visualization filters, and result presentation components within the OsdagBridge Desktop environment.

A.2.2 Architectural Design Contributions

In addition to software development contributions, significant effort was devoted to preparing the High-Level Design (HLD) and Low-Level Design (LLD) documentation for the OsdagBridge Web platform.

The architectural work included:

- React.js and TypeScript frontend architecture.

- Django REST Framework backend architecture.

- PostgreSQL database design.
- Redis and Celery integration planning.
- CAD Scene JSON architecture.
- REST API design specifications.
- Project and workflow management architecture.
- Frontend state management planning.
- Web-based CAD rendering strategy using Konva.

A.2.3 Summary of Contributions

The internship contributions resulted in substantial improvements to the OsdagBridge platform, including desktop application development, CAD visualization capabilities, workspace management, result presentation systems, and future web-platform planning.

The completed work established a foundation for:

- Interactive bridge design visualization.
- Scalable desktop application architecture.
- Real-time parameter synchronization.
- Modular user interface development.
- Browser-based bridge design workflows.
- Future deployment of OsdagBridge as a web platform.

These contributions enhanced the functionality, usability, maintainability, and long-term scalability of the OsdagBridge project.

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.