FOSSEE Winter Internship Report

On

Osdag on Cloud

Submitted by

Samarpita Das

2nd Year B.Tech Student, Department of CSE

Inderprastha Engineering College

Uttar Pradesh

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 12, 2025

Acknowledgments

I would like to express my sincere gratitude to everyone who supported and guided me throughout the course of this project. The journey has been a remarkable experience of learning and growth, and I am truly thankful for this opportunity.

I extend my heartfelt thanks to the Osdag team, including Ajmal Babu M. S., Ajinkya Dahale, and Parth Karia, for their invaluable collaboration and support throughout the project. Their contributions have been instrumental in the success of the work.

Special thanks to the Osdag Principal Investigator (PI), Prof. Siddhartha Ghosh, from the Department of Civil Engineering at IIT Bombay, for his guidance and mentorship, which have been critical in shaping the success of this project. His expertise and advice have played a key role in my personal and professional development.

I would also like to express my gratitude to Prof. Kannan M. Moudgalya, the FOSSEE PI, and FOSSEE Project Investigator, Department of Chemical Engineering at IIT Bombay, for their unwavering support and for creating such a great platform for knowledge exchange and skill enhancement. Their encouragement has been truly inspiring.

I appreciate the guidance and support from FOSSEE Managers Usha Viswanathan, Vineeta Parmar, and their team, whose dedication greatly contributed to the project's success. Their efforts in organizing and supporting the work were invaluable.

I would like to 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 and enabling this project. Their commitment to advancing education through technology has made this project possible.

Lastly, my sincere thanks to my colleagues and teammates, with whom I had the privilege to work closely during this journey.

Contents

Chapter 1

Introduction

1.1 National Mission in Education through ICT

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

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

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

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

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

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

1.1.1 ICT Initiatives of MoE

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

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

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

1.2 FOSSEE Project

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

1.2.1 Projects and Activities

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

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

1.2.2 Fellowships

FOSSEE offers various internship and fellowship opportunities for students:

- Winter Internship
- Summer Fellowship
- Semester-Long Internship

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

For more details, visit the official FOSSEE website.

Figure 1.1: FOSSEE Projects and Activities

1.3 Osdag Software

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

1.3.1 Osdag GUI

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

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

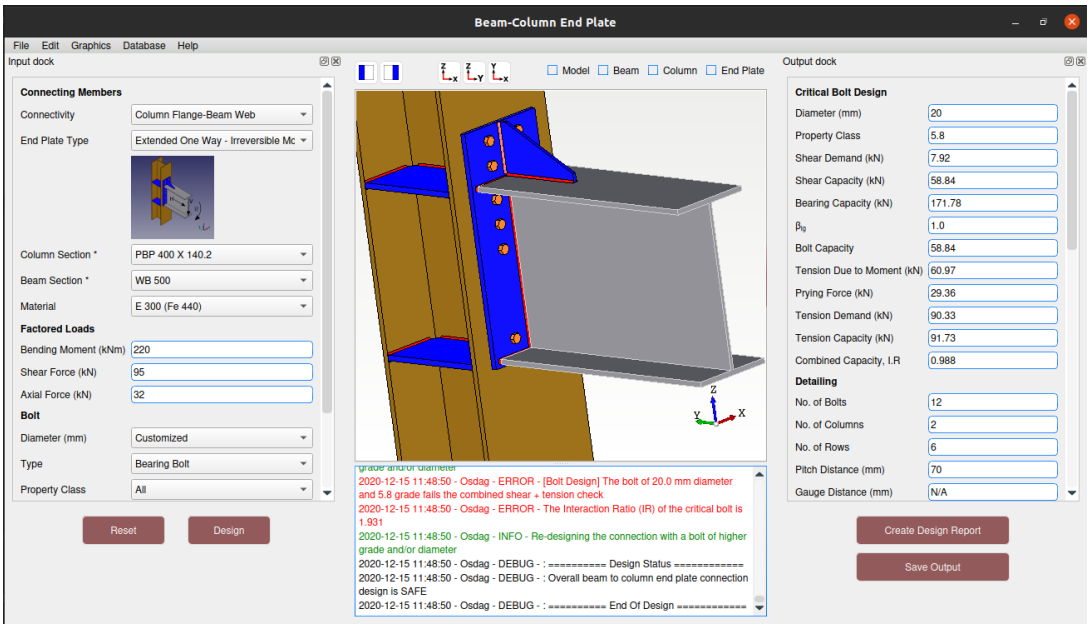

Figure 1.2: Osdag GUI

1.3.2 Features

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

For more details, visit the official Osdag website.

Chapter 2

Screening Task

2.1 Problem Statement

Development of Osdag on cloud project

- Setup and run Osdag on cloud on their device (on the Linux platform)
- Create a UI for Cleat Angle module in Osdag on cloud modules similar to the Osdag-Desktop App
- Develop endpoints for cleat angle identical to the already implemented fin plate and endplate modules

2.2 Tasks Done

The following tasks were performed as part of this screening task:

1. **Setup and Configuration of Osdag on Cloud (Linux Platform)**:
    - Installed and configured the required software and dependencies for running Osdag on a cloud environment, specifically on a Linux platform.
    - Ensured the cloud setup adhered to the system requirements of Osdag, including setting up necessary databases, web services, and ensuring scalability on the cloud.
    - Deployed Osdag in a virtual machine or containerized environment on the cloud to facilitate efficient management, scaling, and maintenance.

2. **User Interface (UI) Creation for Osdag Cloud Modules**:

   - Designed and developed a user interface for the cloud version of Osdag, ensuring that it was visually consistent with the Osdag-Desktop App.
   - Integrated cloud-specific features into the UI, such as user authentication, data storage, and cloud-based processing.
   - Ensured the UI was responsive and intuitive, enabling easy navigation and usage across various devices and screen sizes.
   - Incorporated cloud-specific settings like user roles, permissions, and data management to streamline the use of Osdag in a cloud environment.

3. **Endpoint Development for Cleat Angle Module**:

   - Developed new endpoints for the cleat angle module, following the same architecture as the already existing endpoints for fin plate and endplate modules.
   - Implemented the necessary logic to handle data input, calculations, and output for cleat angle similar to the other modules.
   - Ensured that the new endpoints were fully integrated with the rest of the cloud environment, including cloud storage, databases, and other modules.
   - Performed unit testing and debugging of the cleat angle module endpoints to ensure functionality, accuracy, and performance were at optimal levels.

By completing these tasks, the cloud version of Osdag is now functional, with a user-friendly interface and expanded capabilities to handle cleat angle calculations.

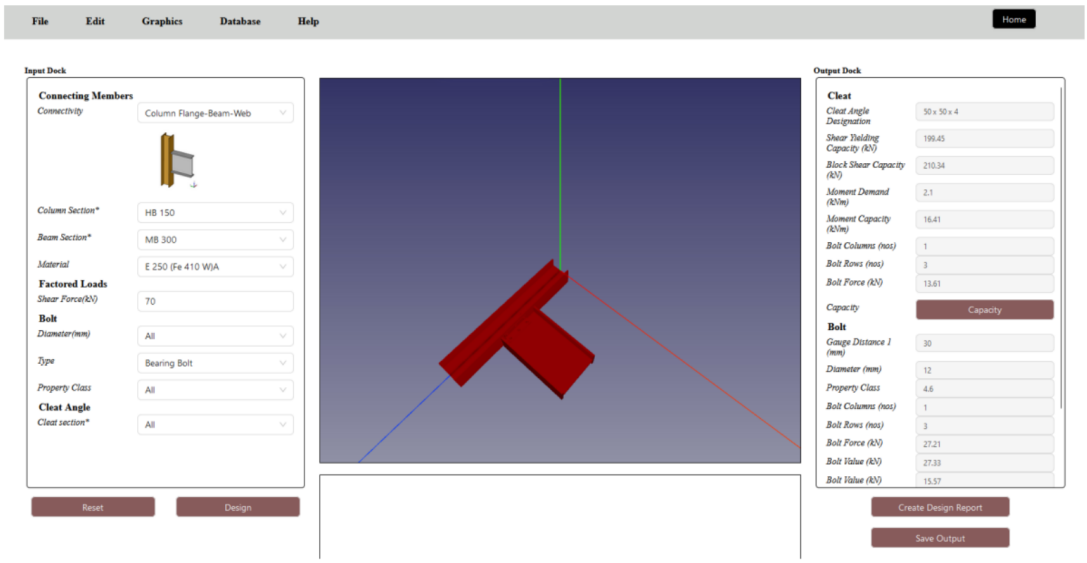

Figure 2.1: Cleat Angle Connection

Chapter 3

Input Data Handling for different Modules

3.1 Problem Statement

changed the method of input data handling in order to successflly handle the needs of different modules such as cleat angle and seated angle

3.2 Tasks Done

created 4 different files in order to handle the input data of different plates according to their need .

3.3 Task 1: Python Code

This section presents a Python script for handling input data for different modules under the Osdag On Cloud. The script is designed to facilitate the process of designing differnt connections by fetching the required data from the database based on user input, ensuring that the user has the necessary information for connection design calculations. If any of the parameters are invalid or missing, it returns error responses with appropriate HTTP status codes.

3.3.1 Description of the Script

The script is structured as follows:

- **Input Parameters**: The user specifies parameters such as connectivity type, bolt diameter, property class, and angle list.
- **Data Fetching**: Based on the input parameters, the script fetches data from various database tables including columns, beams, materials, bolts, and angles. It supports both predefined and custom data, depending on the user's input (e.g., email for custom materials).
- **Connectivity Handling**: The script handles different connectivity scenarios (e.g., 'Column-Flange-Beam-Web', 'Beam-Beam', etc.), returning relevant data such as available column/beam designations and material lists.
- **Customizable Options**: It allows users to customize bolt diameter, property class, and angles. For customized options, it fetches specific data related to bolts, property classes, and angle sizes.

3.3.2 Python Code

The Python script is shown below. Each section is commented for clarity.

Listing 3.1: inputDataview.py for inputing data in Osdag

```
1  %--------------------begin code-------------
2
3  from rest_framework.views import APIView
4  from rest_framework.response import Response
5  from rest_framework import status
6  from rest_framework.parsers import JSONParser
7
8  # Importing models for Columns, Beams, Bolts, and other relevant
       entities
9  from osdag.models import Columns, Beams, Bolt, Bolt_fy_fu, Material,
       CustomMaterials
10 from osdag.models import Design
11
```

```
# Importing input data handlers for different types of connections
from .inputdata.fin_plate_input import FinPlateInputData
from .inputdata.cleat_angle_input import CleatAngleInputData
from .inputdata.end_plate_input import EndPlateInputData
from .inputdata.seated_angle_input import SeatedAngleInputData

# Dictionary to map module names to respective input data handlers
INPUT_DATA_FACTORY = {
    'Fin-Plate-Connection': FinPlateInputData(),
    'Cleat-Angle-Connection': CleatAngleInputData(),
    'End-Plate-Connection': EndPlateInputData(),
    'Seated-Angle-Connection': SeatedAngleInputData(),
}


class InputData(APIView):


    def get(self, request):
        # Extract query parameters from the GET request
        email = request.GET.get("email")
        moduleName = request.GET.get("moduleName")
        connectivity = request.GET.get("connectivity")
        boltDiameter = request.GET.get("boltDiameter")
        propertyClass = request.GET.get("propertyClass")
        thickness = request.GET.get('thickness')
        angleList = request.GET.get('angleList')
        topAngleList = request.GET.get('topAngleList')
        seatedAngleList = request.GET.get('seatedAngleList')
        cookie_id = None

        # Check if the module name exists in the query and print it
        if moduleName is not None:
            print(moduleName)
        else:
            print("module not found")

        # Set cookie_id based on the moduleName received in the request
        if(moduleName == 'Fin-Plate-Connection'):
```

```
51              cookie_id = request.COOKIES.get('
                    fin_plate_connection_session')
52              print('cookie_id inside input data: ', cookie_id)
53          elif(moduleName == 'Cleat-Angle-Connection'):
54              cookie_id = request.COOKIES.get('
                    cleat_angle_connection_session')
55              print('cookie_id inside input data: ', cookie_id)
56          elif(moduleName == 'End-Plate-Connection'):
57              cookie_id = request.COOKIES.get('
                    end_plate_connection_session')
58              print('cookie_id inside end plate input data: ', cookie_id)
59          elif(moduleName == "Seated-Angle-Connection"):
60              cookie_id = request.COOKIES.get('seated_angle_connection')
61              print('cookie id in seated angle connection input data ',
                    cookie_id)
62
63          # Error handling if cookie_id is not found or is empty
64          if cookie_id is None or cookie_id == '':
65              return Response("Error: Please open module", status=status.
                    HTTP_400_BAD_REQUEST)
66
67          # Check if the design session exists in the Design model
68          if not Design.objects.filter(cookie_id=cookie_id).exists():
69              print('The design session does not exists')
70              return Response("Error: This design session does not exist"
                    , status=status.HTTP_404_NOT_FOUND)
71
72          # Check if the module name is valid and exists in the
                INPUT_DATA_FACTORY dictionary
73          if not (moduleName in INPUT_DATA_FACTORY):
74              return Response({"error": "Bad Query Parameter"}, status=
                    status.HTTP_400_BAD_REQUEST)
75
76          # Print email for debugging purposes
77          print("////////////////////////////////////////// ", email)
78
79          # Get the appropriate input data handler for the given module
80          input_data_handler = INPUT_DATA_FACTORY.get(moduleName)
81
```

```
82          # Call the 'process' method of the appropriate input data
                handler
83          return input_data_handler.process(
84              connectivity=connectivity,
85              boltDiameter=boltDiameter,
86              propertyClass=propertyClass,
87              thickness=thickness,
88              angleList=angleList,
89              seatedAngleList=seatedAngleList,
90              topAngleList=topAngleList,
91              email=email
92          )
93
94
95  %-------------------- end code ---------------
```

3.3.3 Explanation of the Code

- Line 1-5: Import necessary classes and functions from rest_framework for handling API requests, responses, and parsing JSON data.
- Line 7-11: Import models for structural components such as Columns, Beams, Bolt, etc., along with the Design model for session tracking.
- Line 13-17: Import input data handlers for different connection types, such as FinPlateInputData, CleatAngleInputData, etc., for processing input data based on module type.
- Line 19-23: Define a dictionary, INPUT_DATA_FACTORY, which maps the connection module names (e.g., 'Fin-Plate-Connection') to their corresponding input data handler classes.
- the rest of the code was written in fin_plate_input.py and other files which is then imported into this file for proper handaling of data

3.3.4 Full code

```
from .input_data_base import InputDataBase
from rest_framework import status
from rest_framework.response import Response
from osdag.models import Columns, Beams, Bolt, Bolt_fy_fu,
   Material, CustomMaterials, Angles
import traceback

class CleatAngleInputData(InputDataBase):
    def process(self, **kwargs):
        connectivity, boltDiameter, angleList = kwargs["
           connectivity"], kwargs["boltDiameter"], kwargs["
           angleList"]
        propertyClass, email  = kwargs["propertyClass"], kwargs["
           email"]

        if (connectivity is None and boltDiameter is None and
           propertyClass is None and angleList is None):
            # fetch the list of all the connectivity options for
               Fin-Plate-Connection
            print("\n\n")
            print('inside connectivtityList handling ')
            print("\n\n")
            connectivityList = ['Column Flange-Beam-Web' , '
               Column Web-Beam-Web', 'Beam-Beam']
            response = {
                'connectivityList': connectivityList
            }
            return Response(response, status=status.HTTP_200_OK)
        if(connectivity == 'Column-Flange-Beam-Web' or
           connectivity == 'Column-Web-Beam-Web'):
            # print('connectivity : ', connectivity)

            try:
                # fetch all records from Column table
```

```
            # fetch all records from Beam table
            # fetch all records from Material table

            columnList = list(Columns.objects.values_list(
                'Designation', flat=True))
            beamList = list(Beams.objects.values_list(
                'Designation', flat=True))

            materialList = list(Material.objects.filter().
               values())
            if email:
                custom_material = list(CustomMaterials.
                   objects.filter(email=email).values())
            materialList = materialList + custom_material

            materialList.append({"id": -1, "Grade": 'Custom'
               })
            response = {
                'columnList': columnList,
                'beamList': beamList,
                'materialList': materialList
            }

            return Response(response, status=status.
               HTTP_200_OK)

        except Exception as err:
            print(err)
            return Response({"error": "Bad request"}, status=
               status.HTTP_400_BAD_REQUEST)

    elif (connectivity == 'Beam-Beam'):
        # print('connectivity : ', connectivity)
```

```
            # fetch all records from Beams table
            # fetch all recorsd from the Material Table
            try:
                beamList = list(Beams.objects.values_list(
                    'Designation', flat=True))
                materialList = list(Material.objects.all().values
                    ())
                materialList.append({"id": -1, "Grade": 'Custom'
                    })
                response = {
                    'beamList': beamList,
                    'materialList': materialList
                }

                return Response(response, status=200)

            except:
                return Response({"error": "Bad request"}, status=
                    status.HTTP_400_BAD_REQUEST)

        elif (boltDiameter == 'Customized'):
            # print('boltDiameter : ', boltDiameter)

            # fetch the data from Bolt table
            try:
                # print('fetching')
                boltList = list(Bolt.objects.values_list(
                    'Bolt_diameter', flat=True))
                boltList.sort()
                print('boltList : ' , boltList)
                response = {
                    'boltList': boltList
                }
```

```
                return Response(response, status=status.
                    HTTP_200_OK)

            except:
                return Response({"error": "Something went wrong"
                    }, status=status.HTTP_400_BAD_REQUEST)

        elif (propertyClass == 'Customized'):
            print('propertyClass : ', propertyClass)

            # fetch the data from Bolt_fy_fu table
            try:
                #boltFyFuList = list(Bolt_fy_fu.objects.
                    values_list(
                #     'Property_Class', flat=True))
                boltFyFuList = ['3.6', '4.6', '4.8', '5.6', '5.8'
                    , '6.8', '8.8', '9.8', '10.9', '12.9']
                # boltFyFuList.sort()

                response = {
                    'propertyClassList': boltFyFuList
                }
                print('propertyFyFuList : ', boltFyFuList)

                return Response(response, status=status.
                    HTTP_200_OK)

            except:
                return Response({"error": "Something went wrong"
                    }, status=status.HTTP_400_BAD_REQUEST)

        elif (angleList == 'Customized'):
            try:
```

```
            # angleList = list(Angles.objects.values_list('
              Designation', flat=True))
            angleList = ['50 x 50 x 3', '50 x 50 x 4', '50 x
              50 x 5', '50 x 50 x 6', '55 x 55 x 4', '55 x
              55 x 5', '55 x 55 x 6', '55 x 55 x 8', '60 x
              60 x 4', '60 x 60 x 5', '60 x 60 x 6', '60 x
              60 x 8', '65 x 65 x 4', '65 x 65 x 5', '65 x
              65 x 6', '65 x 65 x 8', '70 x 70 x 5', '70 x
              70 x 6', '70 x 70 x 8', '70 x 70 x 10', '75 x
              75 x 5', '75 x 75 x 6', '75 x 75 x 8', '75 x
              75 x 10', '80 x 80 x 6', '80 x 80 x 8', '80 x
              80 x 10', '80 x 80 x 12', '90 x 90 x 6', '90 x
               90 x 8', '90 x 90 x 10', '90 x 90 x 12', '100
               x 100 x 6', '100 x 100 x 8', '100 x 100 x 10'
              , '100 x 100 x 12', '110 x 110 x 8', '110 x
              110 x 10', '110 x 110 x 12', '110 x 110 x 16',
               '130 x 130 x 8', '130 x130 x 10', '130 x130 x
               12', '130 x130 x 16', '150 x 150 x 10', '150
              x 150 x 12', '150 x 150 x 16', '150 x 150 x 20
              ', '200 x 200 x 12', '200 x 200 x 16', '200 x
              200 x 20', '200 x 200 x 25', '50 x 50 x 7', '
              50 x 50 x 8', '55 x 55 x 10', '60 x 60 x 10',
              '65 x 65 x 10', '70 x 70 x 7', '100 x 100 x 7'
              , '100 x 100 x 15', '120 x 120 x 8', '120 x
              120 x 10', '120 x 120 x 12', '120 x 120 x 15',
               '130 x 130 x 9', '150 x 150 x 15', '150 x 150
               x 18', '180 x 180 x 15', '180 x 180 x 18', '
              180 x 180 x 20', '200 x 200 x 24']
            response = {
                'angleList': angleList
            }
            return Response(response, status=status.
              HTTP_200_OK)
        except:
```

```
                traceback.print_exc()
                return Response({'error': 'Something went wrong'
                    }, status=status.HTTP_400_BAD_REQUEST)
        return super().process(kwargs)
```

3.4 Task 1: Documentation

3.4.1 Directory Structure

```
OSDAG-WEB
├── inputdata
│   ├── cleat_angle_input.py
│   ├── end_plate_input.py
│   ├── fin_plate_input.py
│   ├── input_data_base.py
│   └── seated_angle_input.py
├── inputDataview.py
└── ...
```

Chapter 4

Module Development: Seated Angle Connection

4.1 Problem Statement

To develop the Seated Angle module for the Osdag on Cloud platform, the goal was to create the necessary UI components, develop new backend endpoints, and integrate the module with the existing backend logic. This would enable the seamless calculation and analysis of seated angle connections within the cloud-based platform.

4.2 Tasks Done

The following tasks were performed as part of the Seated Angle Connection module development:

1. **UI Development for Seated Angle Connection:**
    - Designed and implemented a user interface specifically for the Seated Angle Connection module.
    - Ensured visual consistency with the existing Osdag-Desktop app, while adapting the UI to suit cloud-specific requirements.
2. **Backend Endpoint Development:**

- Developed new backend endpoints for the Seated Angle module, following the same architecture as the existing endpoints for other modules like the fin plate and end plate.
- Implemented logic to handle data input, perform necessary calculations, and output results for the Seated Angle Connection module.

4.3 Outcome

The development and integration of the Seated Angle Connection module successfully enhanced the Osdag on Cloud platform by providing a seamless user interface that adapts to various devices and integrates smoothly with the cloud-based environment. The new backend endpoints were fully functional, ensuring accurate data handling, calculations, and result outputs for seated angle connections. The module was thoroughly tested to ensure its performance, scalability, and stability, thereby expanding the platform's capabilities and offering a comprehensive solution for seated angle connection design and analysis.

Figure 4.1: Seated Angle Connection

Chapter 5

Conclusions

5.1 Tasks Accomplished

The development of the Seated Angle Connection module included designing and implementing a user interface for the cloud platform. Additionally, backend endpoints were created for efficient data handling, calculations, and result generation. The input data handling process was significantly modified to support various modules, ensuring seamless integration and accurate design calculations.

5.2 Skills Developed

Throughout this project, I enhanced my technical skills in React and Django Rest Framework. I gained proficiency in both technologies, along with problem-solving and debugging skills, especially in managing complex design calculations and data integration across modules. Professionally, I also developed skills in task management, workflow prioritization, and team collaboration to achieve project goals.

Chapter A

Appendix

A.1 Work Reports

Name	Samarpita Das
Project	Osdag on cloud
Internship	FOSSEE Winter Internship 2024

Date	Day	Task	Hours spent
12-11-2024	Tuesday	Joining | downloading and setting up Osdag-web	4
13-11-2024	Wednesday	- Merged the ICFOSS fellowship branch into the project. - Finding the issue causing the project to not open after the merge.	4
14-11-2024 15-11-2024 16-11-2024	Thursday Friday Saturday	Resolved merge conflicts, identified dependencies causing issues, and integrated changes.	12
17-11-2024	Sunday	WEEKLY HOLIDAY	
18-11-2024 19-11-2024	Monday	learning about freecad -trying to understand why the freecad model was not opening	9
20-11-2024	Wednesday	handling the cookies and sessions properly	4
21-11-2024 22-11-2024 23-11-2024	Thursday Friday Saturday	rewriting code for cleat angle - input data handling - output generation	12
25-11-2024	Monday	fixing minor bugs in cleat angle connection	4
26-11-2024	Tuesday	Updating the previous input data handling for fin plate	4
27-11-2024	Wednesday	Updating the previous input data handling for end plate	5
28-11-2024 29-11-2024 30-11-2024	Thursday	creating the output dock and output generation for cleat angle	12
02-12-2024	Monday	resolving errors from the client side of cleat angle connection	4
03-12-2024	Tuesday	resolving issue with setting angleList in cleat angle connection	6
04-12-2024	Wednesday	some final changes in cleat angle module	3
05-12-2024	Thursday	creating frontend for seated angle module	4
06-12-2024	Friday	handling session creation and cookie creation for seated angle module	5

07-12-2024 9-12-2024 10-12-2024 11-12-2024	Saturday Monday Tuesday Wednesday	handling angle list in seated angle connection and working on the api endpoints	15
12-12-2024	Thursday	handling angle List in the frontend and also fetching it from the backend in seated angle connection	3
13-12-2024 14-12-2024 16-12-2024	Friday	handling top angle list in seated angle connection	11
17-12-2024	Tuesday	handling top angle List in the frontend and also fetching it from the backend in seated angle connection	2
18-12-2024 - 31-12-2024	Wednesday - Tuesday	connnected the frontend to the backend in seated angle connection processed all the missing elements error made some changes in the pre existing backend in order to effectively handle osdag on cloud created new logs for seated angle connection	30

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.