FOSSEE Winter Internship Report

On

Packaging Osdag creating osdag_latex_env and a New Offline installer

Submitted by

Mehendi Hasan

4th Year B.Sc(Research) Physics Major, Computer Science Minor

Kirori Mal College

University of Delhi

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

February 24, 2026

Abstract

The report documents the work done during a winter internship at FOSSEE, IIT Bombay, focused on packaging Osdag as a conda package and creating an osdag_latex_env for the osdag application. The internship also involved rewriting the offline installer creation process using conda constructor. The report is structured to provide an overview of the project, detailed descriptions of the tasks undertaken, and conclusions drawn from the work. The appendices include additional materials such as code snippets, data, and the work report in PDF format.

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 summarises 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.

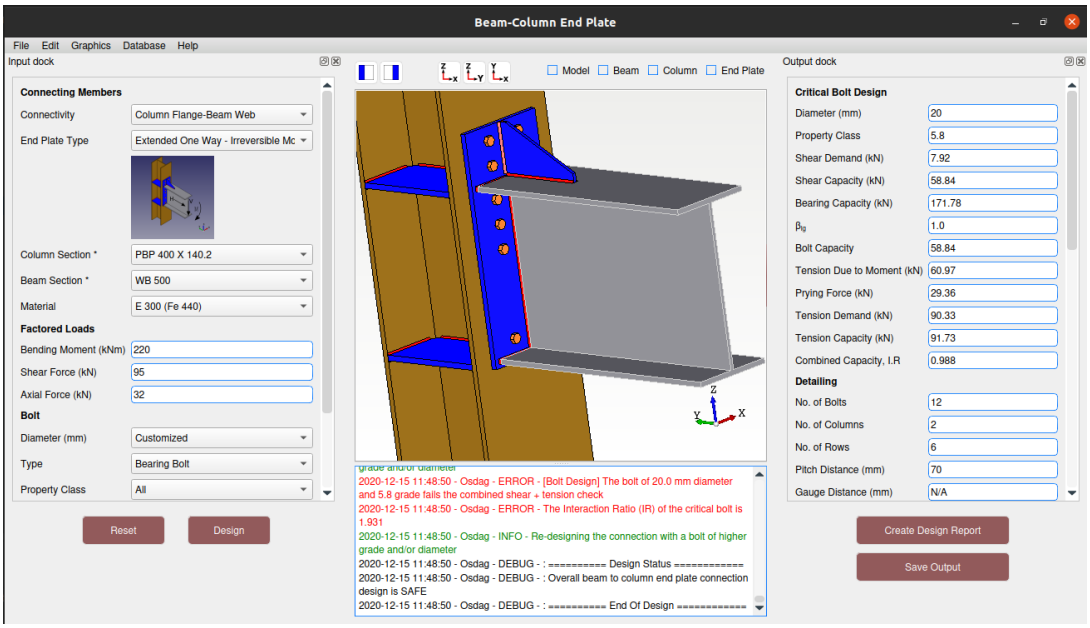

Figure 1.2: Osdag GUI

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: Osdag Latex Env conda package for Osdag

A standalone latex environment for Osdag is required to generate the pdf reports in Osdag. The environment should have all the necessary dependencies installed to compile the latex files. A conda package for Osdag Latec Env was created using the TinyTex distribution of LaTeX. The package includes all the necessary dependencies for compiling the latex files in Osdag. The package can be installed using conda and can be used to generate pdf reports in Osdag without any issues. The package is also compatible with different operating systems, making it easy to use for users on different platforms. The package supports windows, linux_86-64 and further be extended to support other platforms as well.

The package can be installed using the following command:

```
conda install -c osdag osdag-latex-env
```

2.1 Package Creation

The tex distribution was downloaded from TinyTex. The base package for windows, linux and macos was downloaded and extracted separately inside assets folder. The package was created using the conda-build tool. The package recipe was created in a directory called conda-recipe. The recipe includes a meta.yaml file that specifies the package name, version, dependencies, and other metadata. The build section of the ‘meta.yaml‘’ contains the commands to build the package on windows, linux and macos. The package was built

using the conda-build command, and the resulting package was uploaded to the osdag channel on anaconda.org. The package is now available for installation using conda.

2.2 Package Structure

The package structure includes the following files and directories:

```
|-- conda-recipe     # conda recipe
    |-- meta.yaml
|-- osdag_latex_env     # python module
|   |-- __init__.py
|   |-- __main__.py
|   |-- __config__.py
|-- assets       # Tex distribution files for different platforms
|   |   |-- linux
|   |   |   |-- bin/x86_64-linux
|   |   |   |-- texmf-dist/
|   |   |   |-- texmf-var/
|   |   |   |-- texmf-local/
|   |   |   |-- texmf-config/
|   |   |   |-- texmf.cnf
|   |   |   |-- texmfcnf.lua
|   |   |-- win
|   |   |   |-- bin/x86_64-windows
|   |   |   |-- texmf-dist/
|   |   |   |-- texmf-var/
|   |   |   |-- texmf-local/
|   |   |   |-- texmf-config/
|   |   |   |-- texmf.cnf
|   |   |   |-- texmfcnf.lua
|   |   |-- mac/
```

2.3 osdag_latex_env Module

The ‘osdag_latex_env‘ module contains the following files:

```
# __init__.py

from .__main__ import OsdagLatexEnv
```

Listing 2.1: Code File

```
# __main__.py

import os
import shutil
import sys
import platform
import subprocess
from pathlib import Path

class OsdagLatexEnv:
    """
    OsdagLatexEnv provides a lightweight interface to the LaTeX
       toolchain
    installed inside the currently active Conda environment.

    The class does not install or configure LaTeX. Instead, it *
       discovers*
    the existing LaTeX runtime (binaries and data directories) and
        exposes:

    - Paths to LaTeX executables (e.g. pdflatex, bibtex)
    - A registry of all available TeX-related binaries
    - Convenience methods to invoke common tools

    This design treats LaTeX as a system-level toolchain and
       Python
```

```
    as a discovery layer.
    """
    def __init__(self):
        """
        Initialize the LaTeX environment by discovering the active
            Conda
        prefix, detecting the operating system, and scanning for
           available
        LaTeX binaries.
        """
        self.__prefix = Path(sys.prefix)
        self.__system = platform.system().lower()
        self.__machine = platform.machine().lower()
        self.bin_dir = self._detect_bin_dir()
        self.tex_root = self._detect_tex_root()
        self.pdflatex = self._get_pdflatex()

    @property
    def available(self) -> bool:
        """
        Check whether if Module found in this env.

        Returns
        -------
        bool
            True if osdag_latex_env is available, False otherwise.
        """
        if not (self.tex_root and self.pdflatex):
            return False
        try:
            subprocess.run(
                [str(self.pdflatex), "--version"],
                stdout=subprocess.DEVNULL,
                stderr=subprocess.DEVNULL,
```

```
                check=True,
            )
            return True
        except Exception:
            return False

    def _detect_bin_dir(self) -> Path | None:
        """
        Detect the directory containing executables for the
           current platform.

        On Windows (Conda layout):
            <env>/Library/share/osdag_latex_env/bin

        On Linux/macOS:
            <env>/share/osdag_latex_env/bin

        Returns
        -------
        Path | None
            Path to the directory containing LaTeX executables.
        """
        if self.__system == "windows":
            dir = self.__prefix / "Library" / "share" / "
               osdag_latex_env" / "bin" / "x86_64-windows"
            return dir if dir.exists() else None
        elif self.__system == "linux":
            dir = self.__prefix / "share" / "osdag_latex_env" / "
               bin" / "x86_64-linux"
            return dir if dir.exists() else None
        elif self.__system == "darwin":
            dir = self.__prefix / "share" / "osdag_latex_env" / "
               bin" / "universal-darwin"
            return dir if dir.exists() else None
```

```
85
86      def _detect_tex_root(self) -> Path | None:
87          """
88          Detect the root directory containing the LaTeX data tree (
                texmf).
89
90          This is where packages, fonts, and configuration files
                live.
91
92          Returns
93          -------
94          Path
95              Path to the osdag-latex-env texmf root.
96          """
97          if self.__system == "windows":
98              dir = self.__prefix / "Library" / "share" / "
                    osdag_latex_env"
99              return dir if dir.exists() else None
100         else:
101             dir = self.__prefix / "share" / "osdag_latex_env"
102             return dir if dir.exists() else None
103
104     def _get_pdflatex(self) -> Path | None:
105         """
106         Get the path to the pdflatex executable.
107
108         Returns
109         -------
110         Path | None
111             Absolute path to pdflatex, None if not found.
112         """
113         exe = "pdflatex.exe" if self.__system == "windows" else "
                pdflatex"
114         if self.bin_dir and (self.bin_dir / exe).exists():
```

```
115             return self.bin_dir / exe
116         sys_latex = shutil.which("pdflatex")
117         if sys_latex is not None:
118             return Path(sys_latex)
119         return None
```

Listing 2.2: Code File

The module provides utility methods and attribute that can be accessed via OsdagLatexEnv object.

```
1 from pylatex import Document, PageStyle, Head, MiniPage, Foot,
     LargeText, MediumText, LineBreak, simple_page_number, NewPage
2
3 doc = Document(geometry_options=geometry_options, indent=False)
4 latex_env = OsdagLatexEnv()
5 latex_env_present=latex_env.available()
6 if latex_env_present:
7     pdflatex_exec = latex_env.pdflatex
8     doc.generate_pdf(filename, compiler=latex_executable,
         clean_tex = False)
```

Listing 2.3: Example Code

2.4 Tex Distribution

The extracted tex distribution during the download of ‘osdag_latex_env‘ package is extracted into ‘Library/share/osdag_latex_env/‘ in winodws and ‘share/osdag_latex_env/‘ in unix(linux and macos) systems.

2.5 Conde Recipe

The conda-recipe defines how to package ‘osdag_latex_env‘ for each platform. Since the package contains platform specific binaries, it must be packaged on each platform seperately and the .conda file should be uploaded on conda channel(osdag-admin). Currently the package is supported on windows_x86-64, linux_x86-64.

2.5.1 ‘meta.yaml‘

```
package:
  name: osdag_latex_env
  version: "1.0.2"

source:
  path: ..

build:
  number: 0
  noarch: false
  script:

    - {{ PYTHON }} -m pip install . --no-deps -vv

    # Windows
    - mkdir "%PREFIX%\\Library\\share\\osdag_latex_env"      # [win]
    - robocopy "assets\\win" "%PREFIX%\\Library\\share\\osdag_latex_env" /E || IF %ERRORLEVEL% GEQ 8 EXIT /B 1 # [win]

    # Linux
    - mkdir -p "$PREFIX/share/osdag_latex_env"          # [linux]
    - cp -r assets/linux/* "$PREFIX/share/osdag_latex_env/" # [linux]

    # MacOS
    - mkdir -p "$PREFIX/share/osdag_latex_env"     # [osx]
    - cp -r assets/mac/* "$PREFIX/share/osdag_latex_env/" # [osx]


requirements:
```

```
  build:
    - python >=3.11,<=3.13
    - setuptools
    - pip
  host:
    - setuptools
    - pip
    - python >=3.11,<=3.13
  run:
    - python >=3.11,<=3.13

about:
  summary: Self-contained cross-platform LaTeX toolchain for Osdag
  license: LGPL-3.0-only
  license_family: LGPL
  home: https://osdag.fossee.in
  dev_url: https://github.com/osdag-admin/osdag-latex-env
  doc_url: https://github.com/osdag-admin/osdag-latex-env
```

Listing 2.4: Conda-Recipe

2.6 How to build package?

Clone the github repository from osdag-admin/osdag-latex-env. Install conda-build in a conda environment and open repository root folder in terminal.

```
conda install conda-build
cd osdag-latex-env
conda build conda-recipe
```

Upload the .conda file on conda channel. Repeat the step on each platform(windows-64, linux-aarch-64, linux_x86-64, osx_arm-64, osx_arch-64 etc.)

Chapter 3

Internship Task 2: Package Osdag as python package for Conda Distribution

As part of the internship, a Conda distribution package for the refactored latest Osdag was developed to facilitate easy installation and distribution of the software. This task involved creating a Conda recipe, building the package, and uploading it to Anaconda Cloud for public access. The Conda package allows users to install Osdag and its dependencies with a single command, streamlining the setup process for new users and ensuring compatibility across different operating systems. The package includes all necessary files and dependencies required to run Osdag, making it accessible to a wider audience and promoting its adoption in the structural engineering community.

This report details the process of creating the Python package for Osdag, including the organization of the codebase, definition of dependencies, testing procedures, creation of the Conda recipe, and the steps taken to build and distribute the package. The successful completion of this task has resulted in a fully functional Conda package for Osdag that can be easily installed and used by engineers and researchers in the field.

3.1 Codebase Organization

The codebase for osdag was already oragnized in a way that made it easy to package as a python package. The git repository was structured with a clear seperation of source code, data files, and documentation. The source code was organized into modules and submodules, making it easy to identify the main components of the software. The main

entry point for the package was defined in a way that allowed users to easily import and use the functionality of osdag. The data files were also organized in a way that made it easy to include them in the package, ensuring that users had access to all necessary resources when installing the package. Overall, the existing organization of the codebase facilitated the process of creating a Conda package for osdag, allowing for a smooth transition from development to distribution.

```
|-- osdag
|   |-- src
|   |   |-- osdag_core
|   |   |   |-- __init__.py
|   |   |-- osdag_gui
|   |   |   |-- __init__.py
|   |-- conda-recipe
|   |   |-- meta.yaml
|   |-- pyproject.toml
```

3.2 Defining Dependencies

To create a Conda package for Osdag, it was essential to define the dependencies required for the software to function properly. This involved identifying all the external libraries and packages that Osdag relies on, such as NumPy, Pythonocc-core, tbb, smesh etc. The dependencies were specified in a 'meta.yaml' file, which is used by Conda to manage the installation of the package and its dependencies. The 'meta.yaml' file included information about the package name, version, description, and the list of dependencies with their respective versions. By accurately defining the dependencies, we ensured that users would have a seamless experience when installing Osdag through Conda, as all necessary libraries would be automatically installed alongside the package.

The dependencies were carefully selected to ensure compatibility with the latest version of Osdag and to provide users with the necessary tools to utilize the software effectively. Additionally, I made sure to specify the correct versions and channels of the dependencies to avoid any potential conflicts or issues that could arise from incompatible library versions wherever it was needed. This meticulous approach to defining dependen-

cies was crucial in creating a reliable and user-friendly Conda package for Osdag.

3.3 How to Build and Distribute the Package

To build and distribute the Conda package for Osdag, the following steps were taken:

1. First, the ‘pyproject.toml‘ file was created to define the package metadata and dependencies. This file included information such as the package name, version, description, and the non-python data/resource files required for Osdag to function properly.

2. Next, a ‘meta.yaml‘ file was created in the ‘conda-recipe‘ directory to specify the package name, version, description, and the list of dependencies with their respective versions. This file is used by Conda to manage the installation of the package and its dependencies.

3. After defining the necessary files, the package was built using the ‘conda-build‘ command, which compiles the package and its dependencies into a format that can be easily installed by users.

4. Once the package was successfully built, it was uploaded to Anaconda Cloud using the ‘anaconda upload‘ command. This made the package publicly available for users to install and use.

5. Finally, the package was tested by installing it in a new Conda environment and verifying that all functionalities of Osdag were working correctly, ensuring that users would have a seamless experience when installing and using the package through Conda.

```
[build-system]
requires = ["setuptools >= 61.0", "setuptools_scm >= 8.0"]
build-backend = "setuptools.build_meta"
include-package-date = true

[tool.setuptools.packages.find]
namespaces = true
```

```
where = ["src"]

[tool.setuptools.package-data]
"osdag_core.data.doc" = ["**/*.MD", "**/*.md", "**/*.TXT", "**/*.txt"]
"osdag_core.data.ResourceFiles.images" = ["**/*.png", "**/*.PNG", "**/*.jpg", "**/*.JPG", "**/*.jpeg", "**/*.JPEG"]
"osdag_core.data.ResourceFiles.Database" = ["*"]
"osdag_gui.resources" = [
    "**/*.png", "**/*.PNG", "**/*.jpg", "**/*.JPG", "**/*.jpeg", "**/*.JPEG", "**/*.svg",
    "**/*.qss", "**/*.ttf", "**/*.qrc"
]


[project]
name = "osdag"
dynamic = ["version"]
dependencies = ["PyQt5", "PySide6", "requests", "pylatex", "numpy", "PyYaml", "PyGithub", "Click"]
requires-python = ">= 3.11"
description = "Open steel design and graphics"
readme = "README.md"
license = {file = "LICENSE"}
keywords = ["steel design", "civil engineering", "engineering"]

[project.scripts]
osdag = "osdag_gui.__main__:main"
```

Listing 3.1: pyproject.toml file for Osdag 2026.02.0.0 Package

```
# Note: there are many handy hints in comments in this example -- remove them when you've finalized your recipe
# If your package is python based, we recommend using Grayskull to generate it instead:
```

```
# https://github.com/conda-incubator/grayskull

# Jinja variables help maintain the recipe as you'll update the
    version only here.
# Using the name variable with the URL in line 16 is convenient
# when copying and pasting from another recipe, but not really
    needed.
{% set name = "Osdag" %}
{% set branch = "Mergeable" %}
{% set version = "2026.02.0.0" %}

package:
  name: {{ name|lower }}
  version: {{ version }}

source:
  # url: https://github.com/osdag-admin/{{ name }}/archive/refs/
      heads/{{ branch }}.zip
  # git_url: https://github.com/osdag-admin/Osdag.git
  # git_rev: {{ branch }}
  path: ..
  # sha256: ???
  # sha256 is the preferred checksum -- you can get it for a file
      with:
  #   `openssl sha256 <file name>`.
  # You may need the openssl package, available on conda-forge:
  #   `conda install openssl -c conda-forge``

build:
  number: 0
  # Uncomment the following line if the package is pure Python and
       the recipe is exactly the same for all platforms.
  # It is okay if the dependencies are not built for all platforms
      /versions, although selectors are still not allowed.
```

```
31    # See https://conda-forge.org/docs/maintainer/knowledge_base.
          html#noarch-python for more details.
32    noarch: python
33    # If the installation is complex, or different between Unix and
          Windows, use separate bld.bat and build.sh files instead of
          this key.
34    # By default, the package will be built for the Python versions
          supported by conda-forge and for all major OSs.
35    # Add the line "skip: True  # [py<35]" (for example) to limit to
           Python 3.5 and newer, or "skip: True  # [not win]" to limit
          to Windows.
36    # More info about selectors can be found in the conda-build docs
          :
37    # https://docs.conda.io/projects/conda-build/en/latest/resources
          /define-metadata.html#preprocessing-selectors
38    script: {{ PYTHON }} -m pip install --no-deps --force-reinstall
          . -vv
39    entry_points:
40      - osdag = osdag_gui.__main__:main
41
42  requirements:
43    build:
44      # If your project compiles code (such as a C extension) then
            add the required compilers as separate entries here.
45      # Compilers are named 'c', 'cxx' and 'fortran'.
46      - conda-forge::setuptools
47      - conda-forge::wheel
48
49    host:
50      - conda-forge::python>=3.11,<=3.13
51      - conda-forge::setuptools
52      - conda-forge::wheel
53      - conda-forge::pip
54    run:
```

```
55      - conda-forge::python>=3.11,<=3.13
56      - conda-forge::numpy
57      - conda-forge::openpyxl
58      - conda-forge::pandas
59      - conda-forge::pylatex
60      - conda-forge::pyside6
61      - conda-forge::markdown
62      - osdag::osdag_latex_env
63      - conda-forge::pythonocc-core
64      - conda-forge::click
65      - conda-forge::pyyaml
66      - conda-forge::smesh
67      - conda-forge::sqlite
68      - conda-forge::tbb
69
70  test:
71    # Some packages might need a `test/commands` key to check CLI.
72    # List all the packages/modules that `run_test.py` imports.
73    #   imports:
74    #   - simplejson
75    #   - simplejson.tests
76    # For python packages, it is useful to run pip check. However,
        sometimes the
77    # metadata used by pip is out of date. Thus this section is
        optional if it is
78    # failing.
79    #   requires:
80    #     - pip
81    #   commands:
82    #     - pip check
83
84  about:
85    home: https://osdag.fossee.in
86    summary: 'Open steel design and graphics'
```

```
87  description: |
88    Open steel design and graphics
89  # Remember to specify the license variants for BSD, Apache, GPL,
        and LGPL.
90  # Use the SPDX identifier, e.g: GPL-2.0-only instead of GNU
       General Public License version 2.0
91  # See https://spdx.org/licenses/
92  license: LGPL-3.0-only
93  # The license_family, i.e. "BSD" if license is "BSD-3-Clause".
94  # Optional
95  license_family: LGPL
96  # It is required to include a license file in the package,
97  # (even if the license doesn't require it) using the
       license_file entry.
98  # Please also note that some projects have multiple license
       files which all need to be added using a valid yaml list.
99  # See https://docs.conda.io/projects/conda-build/en/latest/
       resources/define-metadata.html#license-file
100 license_file: LICENSE
101 # The doc_url and dev_url are optional.
102 doc_url: https://osdag.fossee.in
103 dev_url: https://github.com/osdag-admin/Osdag
```

Listing 3.2: meta.yaml file for Osdag 2026.02.0.0 Conda Package

3.4 Important Sections while making packages

3.4.1 Conda Recipe

1. The ‘source‘ section is important as it specifies where the source code for the package is located. In this case, it points to the local directory where the Osdag code is located(folder where pyproject.toml file is present), allowing Conda to access the necessary files for building the package.

2. The ‘build‘ section is crucial as it defines how the package should be built and

installed. It specifies that the package is a pure Python package (noarch: python) and provides the command to install the package using pip, without dependencies. Dependencies are handled by conda package manager in the ‘meta,yaml‘ file. Additionally, it defines entry points for the package, allowing users to easily run Osdag from the command line.

3. The ‘requirements‘ section is essential for ensuring that all necessary dependencies are included when installing the package. It lists the build, host, and run dependencies required for Osdag to function properly, ensuring that users have a seamless experience when installing and using the package through Conda.

3.4.2 pyproject.toml

1. The ‘build-system‘ section is important as it specifies the build requirements and backend for the package. It indicates that setuptools is used for building the package and that it requires a minimum version of 61.0, along with setuptools_scm for managing versioning. This ensures that the package can be built correctly and that versioning is handled efficiently.

2. The ‘tool.setuptools.packages.find‘ section is crucial for specifying how setuptools should find the packages to include in the distribution. By setting ‘namespaces‘ to true and defining the ‘where‘ parameter, it allows setuptools to locate the source code in the specified directory, ensuring that all necessary modules are included in the package.

3. The ‘tool.setuptools.package-data‘ section is essential for including non-code files in the package distribution. It specifies which data files should be included in the package, such as documentation files, images, and other resources. This ensures that users have access to all necessary resources when installing the package, enhancing their experience and allowing them to utilize Osdag effectively.

4. The ‘project‘ section is important for defining the metadata of the package, including its name, version, description, and dependencies. This information is crucial for users to understand what the package does and what it requires to function properly. Additionally, the ‘project.scripts‘ section allows for defining command-line

entry points, making it easier for users to run Osdag directly from the terminal.

3.5 Building and Testing the Package

To build the Conda package for Osdag, the ‘conda-build‘ command was used, which compiles the package and its dependencies into a format that can be easily installed by users. The command was executed in the terminal with the appropriate path to the recipe directory, allowing Conda to access the ‘meta.yaml‘ file and build the package according to the specifications defined in it.

After successfully building the package, it was uploaded to Anaconda Cloud using the ‘anaconda upload‘ command, making it publicly available for users to install and use. The package was then tested by creating a new Conda environment and installing the package using the ‘conda install‘ command. After installation, various functionalities of Osdag were verified to ensure that the package was working correctly and that all dependencies were properly installed. This testing process was crucial in ensuring that users would have a seamless experience when installing and using the package through Conda.

3.6 Maintenance and Future Updates

Maintaining the Conda package for Osdag involves regularly updating the package to ensure compatibility with new versions of dependencies and to include any new features or bug fixes. This includes monitoring the dependencies for updates and testing the package with new versions to ensure that it continues to function correctly. Additionally, any changes made to the Osdag codebase should be reflected in the package, requiring updates to the ‘meta.yaml‘ and ‘pyproject.toml‘ files as necessary. It is also important to keep the package documentation up to date, providing users with clear instructions on how to install and use the package, as well as any changes or new features that have been added. Regular maintenance and updates are essential to ensure that the package remains functional and relevant to users, and to promote the continued adoption of Osdag in the structural engineering community.

Chapter 4

Internship Task 3: Packaging Osdag as offline installer.

The Osdag software is a powerful tool for structural engineering design and analysis. To make it more accessible to users, it was packaged as an offline installer. The offline installer allows users to install Osdag without an internet connection, making it easier for users in areas with limited internet access. The packaging process involved creating a standalone executable file that includes all the necessary dependencies for running Osdag. The NSIS installer was created in previous internship, which is used to create the offline installer for Osdag. The installer used a pixi environment to create a virtual environment for Osdag and install all the dependencies inside this environment. The NSIS installer compressed the entired environment as payload and extracted it on user's system along with shortcuts to launch osdag directly from start menu or desktop shortcut.

But the maintanance and updates in this approach is difficult as every time there is an update in osdag, the entire environment needs to be repackaged and redistributed. To overcome this issue, a new approach was taken that used conda's constructor package to make offline installers of conda packages. The Constructor internally uses NSIS to create the installer with a minimal 'constructor.yaml' configuration file. The developer need to only maintain this 'constructor.yaml' file and rest constructor and conda handles on their side. Since Osdag is a conda package, installer with constructor made much more sense than custom NSIS script.

4.1 Repository Structure

```
|-- legacy-installer-setup/
|-- new-constructor-installer/
|   |-- constructor.yaml
|   |-- create_shortcuts.ps1
|   |-- remove_shortcuts.ps1
|   |-- pre_uninstall.bat
|   |-- post_install.bat
|   |-- launch_osdag.vbs
|   |-- launch_osdag.bat
|   |-- license.txt
|   |-- Osdag.ico
```

4.2 Windows Installer Maintanance

The developer need to follow these steps to make new installers for osdag.

- Clone WindowsInstaller repository and navigate to 'new-constructor-installer' directory.
- install constructor in conda environment

```
conda install constructor
```

- update the constructor.yaml file to include updated/new dependencies.
- If header_image and welcoms_image are not provided the constructor makes default images and inserts into installer. This is default behavior which i wanted to avoid so i changed the source code of constructor module, this need to be done if want to replicate exactly same installer UI.
- Go to '**/miniconda3/**/Lib/site-packages/constructor/nsis' folder, open 'main.nsi.tmpl' file comment these lines 197-199 and 203-204 this will remove the default welcome and header images from the installer.

- Then run the constructor command

```
constructor .
```

- upload the generated installer to osdag website and update the download links.

see appendix A for the constructor.yaml file used for creating the installer.

Chapter 5

Conclusion

The internship provided an opportunity to work on various tasks related to the development and maintenance of the Osdag software. The tasks included creating a conda package for a standalone LaTeX environment, redefining the offline installer creation process using conda's constructor package for creating offline installers. The internship also provided an opportunity to learn about different tools and technologies used in software development, such as conda, NSIS, and constructor. Overall, the internship was a valuable learning experience that helped in gaining practical knowledge and skills in software development and maintenance. The experience gained during the internship will be beneficial for future endeavors in the field of software development and engineering.

Chapter A

Constructor Configuration

A.1 constructor.yaml

```
# open ..miniconda3\..\Lib\site-packages\constructor\nsis folder
# open main.nsi.tmpl file comment these lines 197-199 and 203-204
    this will remove the default welcome and header images from the
     installer.


name: Osdag
version: 2026.02.0.0
uninstall_name: ${NAME}
company: Osdag, FOSSEE
installer_type: exe

icon_image: Osdag.ico
welcome_text: |
  Welcome to the Osdag Installer.
  This will install Osdag with all required runtime components.

license_file: ..\license.txt
extra_files:
  - post_install.bat
  - pre_uninstall.bat
```

```
  - launch_osdag.vbs
  - Osdag.ico
  - create_shortcuts.ps1
  - remove_shortcuts.ps1



channels:
  - conda-forge
  - osdag

specs:
  - osdag::osdag=2026.02.0.0

# Add in system env variables
initialize_conda: false
register_python: false

# Set Default installation location
default_prefix: "%LOCALAPPDATA%\\Osdag"
default_prefix_all_users: "%ProgramData%\\Osdag"

# if package conda recipe include has Menu\ then use for shortcut
    creation
# menu_packages:
#   - osdag

post_install: post_install.bat
pre_uninstall: pre_uninstall.bat

post_install_desc: |
 This will create shortcuts for Osdag and add it to the start menu
    . You can also launch Osdag from the desktop shortcut.(
    Recommended)
```

```
51
52
53 conclusion_text: |
54   Osdag has been successfully installed.
55   You can launch Osdag from the start menu or desktop shortcut. To
       uninstall Osdag, please go to the Control Panel and remove
     it from the list of installed programs. If you have any
     issues, please contact https://osdag.fossee.in.
```

A.2 post_install.bat

```
1 @echo off
2 setlocal
3
4 "%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -
    NoProfile -ExecutionPolicy Bypass -File "%PREFIX%\
    create_shortcuts.ps1" -InstallDir "%PREFIX%"
5
6
7 endlocal
8 exit /b 0
```

A.3 create_shortcuts.ps1

```
1 param(
2     [string]$InstallDir
3 )
4
5 $AppName = "Osdag"
6 $Version = "2026.02.0.0"
7 $AppExe = Join-Path $InstallDir "Scripts\osdag.exe"
8 $Launcher = Join-Path $InstallDir "launch_osdag.vbs"
9 $UninstallExe = Join-Path $InstallDir "Uninstall-$AppName.exe"
```

```
$IconFile = Join-Path $InstallDir "Osdag.ico"

# If user is admin?
$currentUser = [Security.Principal.WindowsIdentity]::GetCurrent()
$principal = New-Object Security.Principal.WindowsPrincipal(
    $currentUser)
$isAdmin = $principal.IsInRole([Security.Principal.
    WindowsBuiltInRole]::Administrator)
if ($isAdmin) {
    $StartMenuDir = Join-Path $env:ProgramData "Microsoft\Windows\
        Start Menu\Programs\$AppName"
    $DesktopPath = Join-Path $env:Public "Desktop"
}else {
    $StartMenuDir = Join-Path $env:AppData "Microsoft\Windows\
        Start Menu\Programs\$AppName"
    $DesktopPath = [Environment]::GetFolderPath("Desktop")
}


if (!(Test-Path $StartMenuDir)) {
    New-Item -ItemType Directory -Path $StartMenuDir | Out-Null
}
if (!(Test-Path $DesktopPath)) {
    New-Item -ItemType Directory -Path $DesktopPath | Out-Null
}

$Shell = New-Object -ComObject WScript.Shell

# ---- Desktop Shortcut ----
$Shortcut = $Shell.CreateShortcut("$DesktopPath\$AppName.lnk")
$Shortcut.TargetPath = $Launcher
$Shortcut.WorkingDirectory = "$InstallDir"
$Shortcut.IconLocation = "$IconFile,0"
$Shortcut.Save()
```

```
40
41 # ---- App Shortcut (Start Menu) ----
42 $Shortcut = $Shell.CreateShortcut("$StartMenuDir\$AppName.lnk")
43 $Shortcut.TargetPath = $Launcher
44 $Shortcut.WorkingDirectory = $InstallDir
45 $Shortcut.IconLocation = "$IconFile,0"
46 $Shortcut.Save()
47
48
49 # ---- Write Registry Entry for Uninstall ----
50 $SizeBytes = (Get-ChildItem $InstallDir -Recurse -Force | Measure-
       Object -Property Length -Sum).Sum
51 $SizeKB = [math]::Round($SizeBytes / 1KB)
52
53 if ($isAdmin) {
54
55     $baseKey = [Microsoft.Win32.RegistryKey]::OpenBaseKey(
56         [Microsoft.Win32.RegistryHive]::LocalMachine,
57         [Microsoft.Win32.RegistryView]::Registry64
58     )
59
60     $subKey = $baseKey.CreateSubKey(
61         "Software\Microsoft\Windows\CurrentVersion\Uninstall\
               $AppName"
62     )
63
64     $subKey.SetValue("DisplayName", $AppName)
65     $subKey.SetValue("UninstallString", "`"$UninstallExe`"")
66     $subKey.SetValue("DisplayIcon", $IconFile)
67     $subKey.SetValue("Publisher", "Osdag, FOSSEE")
68     $subKey.SetValue("DisplayVersion", $Version)
69     $subKey.SetValue("InstallLocation", $InstallDir)
70     $subKey.SetValue("HelpLink", "https://osdag.fossee.in/")
```

```
    $subKey.SetValue("EstimatedSize", $SizeKB, [Microsoft.Win32.
        RegistryValueKind]::DWord)
    $subKey.Close()
    $baseKey.Close()

}
else {

    $UninstallKey = "HKCU:\Software\Microsoft\Windows\
        CurrentVersion\Uninstall\$AppName"

    New-Item -Path $UninstallKey -Force | Out-Null
    Set-ItemProperty -Path $UninstallKey -Name "DisplayName" -
        Value $AppName
    Set-ItemProperty -Path $UninstallKey -Name "UninstallString" -
        Value "`"$UninstallExe`""
    Set-ItemProperty -Path $UninstallKey -Name "DisplayIcon" -
        Value $IconFile
    Set-ItemProperty -Path $UninstallKey -Name "Publisher" -Value
        "Osdag, FOSSEE"
    Set-ItemProperty -Path $UninstallKey -Name "DisplayVersion" -
        Value $Version
    Set-ItemProperty -Path $UninstallKey -Name "InstallLocation" -
        Value $InstallDir
    Set-ItemProperty -Path $UninstallKey -Name "HelpLink" -Value "
        https://osdag.fossee.in/"
    New-ItemProperty -Path $UninstallKey -Name "EstimatedSize" -
        Value $SizeKB -PropertyType DWord -Force | Out-Null
}
```

A.4 launch_osdag.vbs

```
Set objShell = CreateObject("Wscript.Shell")
```

```
root = CreateObject("Scripting.FileSystemObject").
    GetParentFolderName(WScript.ScriptFullName)

cmd = "cmd /c ""set PATH=" & root & "\Library\bin;" & root & "\
    Scripts;%PATH% && """ & root & "\Scripts\osdag.exe"""""

objShell.Run cmd, 0
```

A.5 pre_uninstall.bat

```
@echo off
setlocal

"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -
    NoProfile -ExecutionPolicy Bypass -File "%PREFIX%\
    remove_shortcuts.ps1" -InstallDir "%PREFIX%"

endlocal
exit /b 0
```

A.6 remove_shortcuts.ps1

```
$AppName = "Osdag"

# If user is admin?
$currentUser = [Security.Principal.WindowsIdentity]::GetCurrent()
$principal = New-Object Security.Principal.WindowsPrincipal(
    $currentUser)
$isAdmin = $principal.IsInRole([Security.Principal.
    WindowsBuiltInRole]::Administrator)
if ($isAdmin) {
     $StartMenuDir = Join-Path $env:ProgramData "Microsoft\Windows\
        Start Menu\Programs\$AppName"
```

```
9       $DesktopPath = Join-Path $env:Public "Desktop"
10      $UninstallKey = "HKLM:\Software\Microsoft\Windows\
            CurrentVersion\Uninstall\$AppName"
11  }else {
12      $StartMenuDir = Join-Path $env:AppData "Microsoft\Windows\
            Start Menu\Programs\$AppName"
13      $DesktopPath = [Environment]::GetFolderPath("Desktop")
14      $UninstallKey = "HKCU:\Software\Microsoft\Windows\
            CurrentVersion\Uninstall\$AppName"
15  }
16
17  Remove-Item "$StartMenuDir" -Recurse -Force -ErrorAction
        SilentlyContinue
18  Remove-Item "$DesktopPath\$AppName.lnk" -Force -ErrorAction
        SilentlyContinue
19  Remove-Item $UninstallKey -Recurse -Force -ErrorAction
        SilentlyContinue
```

Date	DAY	TASK	Hours worked
1 Dec 2025	Monday	Make Input Dock lock state more visible	4
2 Dec 2025	Tuesday	Make Input Dock lock state more visible ,Change un	4
3 Dec 2025	Wednesday	Resize HomePage icons	2
4 Dec 2025	Thursday	Comment out unneccessary print statements, Fix va	4
5 Dec 2025	Friday	Add input dock unlock clears output dock feature	4
8 Dec 2025	Monday	Merge Interns work, work with suhail	4
9 Dec 2025	Tuesday	Merge Interns work, work with suhail	4
7 Jan 2026	Wednesday	Look for new release of osdag	4
8 Jan 2026	Thursday	Look for osdagBridge integration as plugin, and crea	6
9 Jan 2026	Friday	check issue and close on github	4
12 Jan 2026	Monday	Refactor CLI module	4
13 Jan 2026	Tuesday	cli module onclude new modules, work on issuse	5
14 Jan 2026	Wednesday	bug fixes	4
15 Jan 2026	Thursday	bug fixes	4
16 Jan 2026	Friday	Fix relative path	6
19 Jan 2026	Monday	remove latex env	4
20 Jan 2026	Tuesday	Work on osdagBridge as plugin	4
21 Jan 2026	Wednesday	Work on osdagBridge as plugin	4
22 Jan 2026	Thursday	osdag 2026.01.0.0 release packaging and final comi	4
23 Jan 2026	Friday	osdag 2026.01.0.0 release packaging and final comi	4
24 Jan 2026	Saturday	refactor plugin system to match current changes	6
25 Jan 2026	Sunday	osdag 2026.01.0.0 release packaging and final comi	5
26 Jan 2026	Monday	osdag 2026.01.0.0 release packaging and final comi	4
27 Jan 2026	Tuesday	osdag 2026.01.0.0 release packaging and final comi	5
28 Jan 2026	Wednesday	osdag 2026.01.0.0 release packaging and final comi	6
29 Jan 2026	Thursday	Prepare osdag-latex-env as conda package	5
30 Jan 2026	Friday	Prepare osdag-latex-env as conda package	8
2 Feb 2026	Monday	osdag 2026.01.0.0 release packaging	5
3 Feb 2026	Tuesday	osdag 2026.01.0.0 release packaging	4
4 Feb 2026	Wednesday	Package osdag_latex_env	7
5 Feb 2026	Thursday	osdag 2026.02.0.0 release packaging	8
6 Feb 2026	Friday	osdag 2026.02.0.0 release packaging	8
7 Feb 2026	Saturday	Create Installer for new release	4
8 Feb 2026	Sunday	Create Installer for new release	4
9 Feb 2026	Monday	Make installer using constructor package, no pixi en	5
10 Feb 2026	Tuesday	Make installer using constructor package, no pixi en	5
11 Feb 2026	Wednesday	Make installer using constructor package, no pixi en	4

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.