



# FOSSEE Semester Long Internship Report

On

Development of an Installer, Tex Manager and  
LaTeX Testing Suite for Osdag

Submitted by

**Steve Sojan**

*4th Year B.Tech Student, School of Computing Science Engineering and Artificial  
Intelligence*

*Vellore Institute of Technology*

*Bhopal*

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

June 23, 2025

# Acknowledgments

- I would like to express my sincere gratitude to the Osdag and FOSSEE team at IIT Bombay for providing me with this valuable opportunity. This internship has been an enriching experience, allowing me to expand my knowledge and gain meaningful insights.
- First and foremost, I extend my sincere thanks to the project staff of the Osdag team, including Ajmal Babu M. S., Ajinkya Dahale, and Parth Karia, for their unwavering support, insights, and expertise throughout the project. Their dedication and guidance have greatly enriched my learning experience.
- I am deeply grateful to Prof. Siddhartha Ghosh, Principal Investigator of the Osdag project and a faculty member in the Department of Civil Engineering at IIT Bombay, for his vision, mentorship, and encouragement. His leadership has been instrumental in shaping the success of this project.
- I would also like to acknowledge the invaluable support and guidance of Prof. Kannan M. Moudgalya, Principal Investigator of the FOSSEE Project, Department of Chemical Engineering, IIT Bombay. His passion for fostering open-source initiatives has been a source of inspiration for me.
- My sincere thanks to the FOSSEE managers, Usha Viswanathan and Vineeta Parmar, along with their entire team, for providing exceptional support and creating an environment conducive to learning and growth.
- This project would not have been possible without the support of the National Mission on Education through Information and Communication Technology (ICT), Ministry of Education (MoE), Government of India. Their commitment to advancing open source software for education has made this opportunity possible and has

driven innovation for millions.

- I also extend my gratitude to my colleagues who worked alongside me during this internship. Their collaboration, ideas, and inputs made the journey all the more enjoyable and productive.
- I would like to thank my college - Vellore Institute of Technology, my Department Program Chair, Dr. Rajit Nair for promptly directing me to the available opportunity and my esteemed professors for supporting me throughout this journey.

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	National Mission in Education through ICT . . . . .	5
1.1.1	ICT Initiatives of MoE . . . . .	6
1.2	FOSSEE Project . . . . .	7
1.2.1	Projects and Activities . . . . .	7
1.2.2	Fellowships . . . . .	7
1.3	Osdag Software . . . . .	8
1.3.1	Osdag GUI . . . . .	9
1.3.2	Features . . . . .	9
<b>2</b>	<b>Screening Task</b>	<b>10</b>
2.1	Problem Statement . . . . .	10
2.2	Tasks Done . . . . .	10
<b>3</b>	<b>Internship Task 1</b>	
	<b>Osdag LaTeX Environment</b>	<b>19</b>
3.1	Task 1: Problem Statement . . . . .	19
3.2	Task 1: Tasks Done . . . . .	19
3.3	Task 1: Significance of each branch of the texmf tree . . . . .	20
3.4	Task 1: Python Code . . . . .	21
3.4.1	Description of the Script . . . . .	21
3.4.2	Python Code . . . . .	21
3.4.3	Explanation of the Code in Initializing Environment Variables Snippet . . . . .	22
3.4.4	Explanation of the Code Snippet added to SectionModeller_Latex.py	22
3.4.5	Explanation of the Code Snippet added to reportGenerator_latex.py	23
<b>4</b>	<b>Internship Task 2</b>	
	<b>Tex Manager -Maintainer's Workflow</b>	<b>25</b>
4.1	Task 2 Problem Statement . . . . .	25
4.2	Task 2: Workflow . . . . .	25

4.3	Task 2: Setting Up Environment for the Maintainer . . . . .	27
4.4	Task 2: Implementing Git Submodules . . . . .	29
4.5	Task 2: To view all available commits in the Submodule . . . . .	30
4.6	Task 2: To change the commit hash referenced . . . . .	30
<b>5</b>	<b>Internship Task 3</b>	
	<b>LaTeX Testing Suite</b>	<b>33</b>
5.1	Task 3 Problem Statement . . . . .	33
5.2	Task 3: Arrange, Act and Assert Approach . . . . .	33
5.3	Task 3: Checks Done . . . . .	34
5.4	Task 3: Helper Functions . . . . .	34
5.5	Task 3: Tests Conducted . . . . .	35
5.6	Task 3: Report Generation System . . . . .	44
<b>6</b>	<b>Conclusions</b>	<b>46</b>
6.1	Tasks Accomplished . . . . .	46
6.2	Skills Developed . . . . .	47
<b>A</b>	<b>Appendix</b>	<b>49</b>
A.1	Work Reports . . . . .	49
	<b>Bibliography</b>	<b>52</b>

# 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](http://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
<b>Audio-Video e-content</b>			
1	SWAYAM	Earn credit via online courses	Develop and host courses; accept credits
2	SWAYAMPBABHA	Access 24x7 TV programs	Enable SWAYAMPBABHA viewing facilities
<b>Digital Content Access</b>			
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
<b>Hands-on Learning</b>			
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
<b>E-Governance</b>			
11	SAMARTH ERP	Manage student lifecycle digitally	Enable institutional e-governance
<b>Tracking and Research Tools</b>			
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.

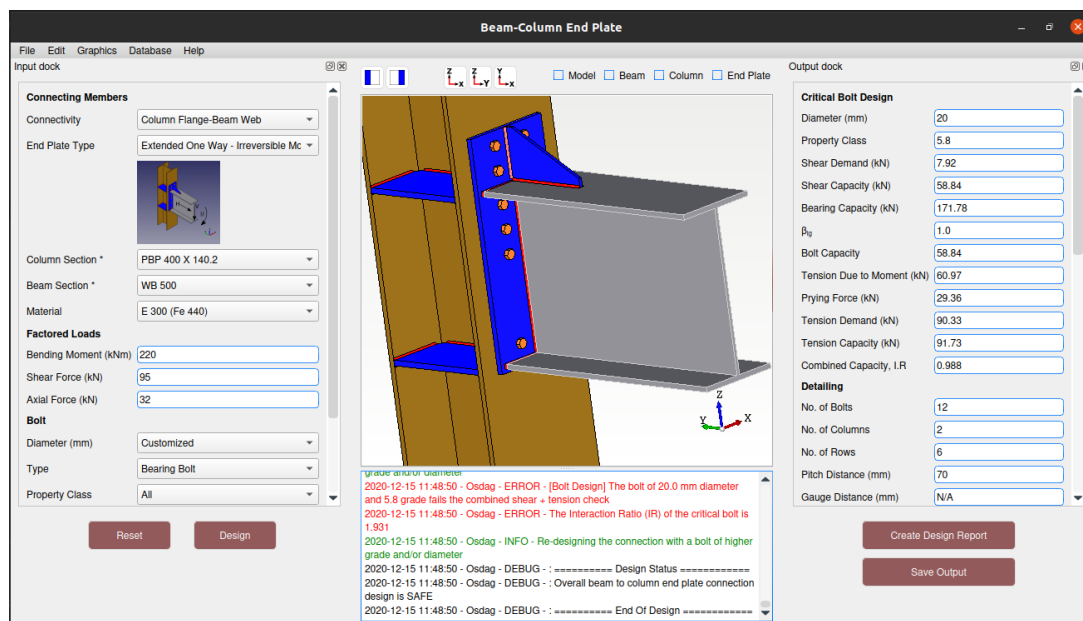


Figure 1.2: Osdag GUI

### 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

Creating a Windows Installer for the Osdag Software

### 2.2 Tasks Done

Github Repo for Screening Task - <https://github.com/stevesojan/Osdag-exe>

Inno Setup extracts the base installation files in a directory which the user selects. These installation files run automatically, to programmatically install Osdag, complete with a Startup menu icon and Desktop shortcut.

My approach was to break the problem down into smaller components, as to what and how I would have to install in a certain order. This I figured out with the steps as mentioned on Osdag's official website. It boiled down to 3 steps: TinyTex Miniconda Osdag(using conda)

#### **TinyTex**

Essentially it came down to using alternatives that were lightweight, which was a core part of the problem statement of the project. With due research, I found TinyTex, a lightweight alternative to MikTex. The script always downloads the latest available version of TinyTex-1 which is updated as a daily build, from GitHub.

.bat file from the TinyTex Official Release Page: <https://yihui.org/tinytex/#installation>

To install TinyTex as mentioned on its official release page, it offers a .bat file which

when run extracts the files from the TinyTex release on GitHub into the user's App-Data/Roaming folder.

This was a challenging bit to work on, because the original .bat file employs 3 methods to install TinyTex and Method 1 (System.Net.Http) in the order of execution was the least stable method. System.Net.Http is a low level, more modern API, powerful but far less stable compared to System.Net.WebClient which is a high level API, being used since 2002. Also, System.Net.Http is used for modern .NET frameworks, i.e it could work well in newer Win systems but might fail to work in older Win systems. Ideally the far more stable choice was System.net.WebClient.

Listing 2.1: Python Code for TinyTex Installation

```
1 %-----begin code-----
2 import os
3 import subprocess
4 def install_tinytex():
5     """Installs TinyTeX and configures it, with fallback download
6         methods."""
7     try:
8         print("Installing TinyTeX...")
9
10        # Step 1: Define Environment Variables
11        tinytex_url = "https://github.com/rstudio/tinytex-releases/
12                      releases/download/
13                      daily/TinyTeX-1.zip"
14        temp_dir = os.getenv("TEMP")
15        tinytex_dir = os.getenv("APPDATA", "Roaming")
16        downloaded_file = os.path.join(temp_dir, "install.zip")
17
18        # Step 2: Clean up Temporary Files
19        print("Cleaning up any leftover TinyTeX directories...")
20        tinytex_pattern = os.path.join(tinytex_dir, "TinyTeX*")
21        subprocess.run(f'del /f /s /q "{tinytex_pattern}" & rmdir /s /q
22                      "{tinytex_dir}\\TinyTeX"', shell=True,
23                      check=False)
24
25        # Step 3: Download TinyTeX with Multiple Methods
26        success = False
```

```

24     print("Downloading TinyTeX...")
25
26     # Method 1: System.Net.WebClient
27     print("Method 1: Using System.Net.WebClient...")
28     print(
29         "Do not panic if you see text in red, or if the shell looks
30         stuck, the installer is working, please wait ;)")
31     download_command = f'powershell -Command "[Net.
32         ServicePointManager]::SecurityProtocol = [Net.
33         SecurityProtocolType]::Tls12; (New-Object System.Net.
34         WebClient).DownloadFile(\'{tinytex_url}\', \'{
35         downloaded_file}\')"'
36     if subprocess.run(download_command, shell=True).returncode ==
37         0:
38         success = True
39
40     # Method 2: Invoke-WebRequest
41     if not success:
42         print("Method 1 failed. Trying Method 2: Invoke-WebRequest
43         ...")
44         download_command = f'powershell -Command "[Net.
45         ServicePointManager]::SecurityProtocol = [Net.
46         SecurityProtocolType]::Tls12; Invoke-WebRequest \'{
47         tinytex_url}\' -OutFile \'{downloaded_file}\''
48         if subprocess.run(download_command, shell=True).returncode
49             == 0:
50             success = True
51
52     # Method 3: System.Net.Http.HttpClient
53     if not success:
54         print("Method 2 failed. Trying Method 3: System.Net.Http.
55         HttpClient...")
56         download_command = f'powershell -Command "& {{ try {{ Add-
57         Type -A \'System.Net.Http\'; [Net.ServicePointManager]::
58         SecurityProtocol = [Net.SecurityProtocolType]::Tls12;
59         $response = (New-Object System.Net.Http.HttpClient).
60         GetAsync(\'{tinytex_url}

```

```

47
48
49         }\'); $response.Wait(); $outputFileStream = [System.IO.
        FileStream]::new(\'{downloaded_file}\', [System.IO.
        FileMode]::Create, [System.IO.FileAccess]::Write);
        $response.Result.Content.CopyToAsync($outputFileStream).
        Wait(); $outputFileStream.Close() }} catch {{ throw $_
        }} }}"'
50     if subprocess.run(download_command, shell=True).returncode
        == 0:
51         success = True
52
53     # Exit if all download methods fail
54     if not success:
55         print("All download methods failed. Unable to download
        TinyTeX.")
56         return
57
58     # Step 4: Unzip the File
59     print("Unzipping TinyTeX...")
60     unzip_command = f'powershell -Command "& {{ Add-Type -A \'
        System.IO.Compression.FileSystem\'; [System.IO.Compression.
        ZipFile]::ExtractToDirectory(\'{downloaded_file}\', \'{
        temp_dir}\'); }}"'
61     subprocess.run(unzip_command, shell=True, check=True)
62     os.remove(downloaded_file)
63
64     # Step 5: Move TinyTeX to Target Directory
65     print(f"Moving TinyTeX to {tinytex_dir}...")
66     tinytex_path = os.path.join(tinytex_dir, "TinyTeX")
67     subprocess.run(f'move /y "{os.path.join(temp_dir, "TinyTeX")}"
        "{tinytex_path}"', shell=True, check=True)
68
69     # Step 6: Configure tlmgr
70     print("Configuring TinyTeX...")
71     tlmgr_path = os.path.join(tinytex_path, "bin", "windows", "
        tlmgr")
72     subprocess.run(f'"{tlmgr_path}" path add', shell=True, check=
        True)

```

```

73     subprocess.run(f'"{tlmgr_path}" option repository ctan', shell=
        True, check=True)
74     subprocess.run(f'"{tlmgr_path}" postaction install script xetex
        ', shell=True, check=True)
75
76     print("TinyTeX installed and configured successfully.")
77     print("Now Installing additional required packages...")
78     subprocess.run(
79         [f'{tlmgr_path}', 'install', 'lastpage', 'parskip', '
            needspace', 'fancyhdr', 'colortbl', 'multirow'],
80         check=True, shell=True)
81     print("Required Packages Installed Successfully")
82
83     except subprocess.CalledProcessError as e:
84         print(f"An error occurred during TinyTeX installation: {e}")
85
86
87 install_tinytex()

```

In testing the .bat file directly, Method 1(System.Net.Http) used to most always fail and exit the shell abruptly, and only when the user invoked the install-bin-windows.bat file again would it try Method 2. So I redesigned the logic and invoked System.Net.WebClient as Method 1, Method -2 to Invoke Web-Requests and Method-3 to System.Net.Https as fallback mechanisms in a new batch file I created “tinytex\_bin\_steve.bat” present in the repo.

The respective method (generally System.Net.WebClient API) downloads the daily build(latest version) of TinyTex-1(currently file size of 110 MB) from the url stored in the variable "tinytex\_url", which is subsequently used by the respective API to download, unzip and store in a temp directory. In the temp directory after unzipping, the downloaded zip file is deleted and all contents of the temp directory are moved to App-Data/Roaming/TinyTex.

Now comes configuring TinyTex, i.e adding tlmgr to the system path and directing tlmgr to use the Comprehensive TeX Archive Network (CTAN) mirror as the source for downloading or updating TeX packages.

As for the missing packages, necessary for Osdag, I test generated every design report which would also generate a .tex file, essential in figuring out which package was missing

by cross-checking with the `\usepackage` flag in the `.tex` file.

## Miniconda

For Miniconda installation, I preferred the script to always download the latest Miniconda installer, available for Win 64 bit systems, from their official release page(mentioned below). By doing this, for future installations, the installer would not be pre - bundled with a version which could become obsolete in the future. But for users who were existing Miniconda users, would also possibly have everything configured, including other environments within their existing miniconda installation, the script checks if there is an existing installation. If there isn't, only then will it proceed to install a fresh, latest version of Miniconda. This was achieved by using python's requests module and the repo which releases Miniconda's latest version : <https://repo.anaconda.com/miniconda/>

Since the file size of the Miniconda setup was 81.7 MB, I set `stream = True`, which signals the requests module to download the file in a stream and not all at once, and thus allows for lesser RAM to be used. For computers with lesser memory, this would be a boon. The file is downloaded in a stream of chunks, with each chunk, 1024 bytes/1KB in size.

Listing 2.2: Python Code for Miniconda Installation

```
1  %-----begin code-----
2  import os
3  import subprocess
4  import requests
5  import time
6
7  def install_miniconda():
8      upf = os.environ.get('USERPROFILE')
9      if os.path.exists(r"C:\miniconda3") == False and os.path.exists(f"{
      upf}\\miniconda3") == False:
10         # os.path.exists(r"C:\miniconda3") == False - checks if
            miniconda3 is not installed system wide for all users in C
            drive
11         installer_url = "https://repo.anaconda.com/miniconda/Miniconda3
            -latest-Windows-x86_64.exe"
12         installer_name = "Miniconda3-latest-Windows-x86_64.exe"
13
14
```



```

15
16     print("Downloading the latest Miniconda installer...")
17     response = requests.get(installer_url, stream=True)
18     with open(installer_name, "wb") as file:
19         for chunk in response.iter_content(chunk_size=1024):
20             if chunk: # Filter out keep-alive chunks
21                 file.write(chunk)
22             print("Download completed.")
23
24     print("Installing Miniconda...")
25     subprocess.run(["start", "/wait", "Miniconda3-latest-Windows-
        x86_64.exe", "/InstallationType=JustMe", "/RegisterPython=0"
        , "/AddToPath=1", "/S"], shell=True)
26     print("Miniconda Installed Successfully")
27 else:
28     print("Miniconda is already installed.")

```

## Osdag

Osdag also installs as a conda package, but I have also designed a start menu button and Desktop shortcut, which are available post installation. Installing Osdag through the conda channel, ensures the latest version is being installed and a feature can be integrated where prior users get the choice to update to the latest version.

Listing 2.3: Python Code for Osdag Installation

```

1  %-----begin code-----
2  def create_conda_env():
3      """Creates the Conda environment and installs Osdag in the same
        shell."""
4      upf = os.environ.get("USERPROFILE")
5      conda_bat = os.path.join(upf, "miniconda3", "condabin", "conda.bat"
        )
6
7      # Check if conda.bat exists
8      if not os.path.exists(conda_bat):
9          raise FileNotFoundError(f"Conda was not found at {conda_bat}.
        Ensure Miniconda is installed.")
10
11     print("Creating Conda environment and installing Osdag...")
12

```

```

13     success = False
14     attempt = 1
15
16     while not success and attempt<=10:
17         try:
18             print(f"Attempt {attempt}: Running Conda command to install
19                 Osdag...")
20
21             # Run the Conda command in the same shell
22             subprocess.run(
23                 f'"{conda_bat}" activate && conda create -n osdag-env
24                 osdag::osdag -c conda-forge -y',
25                 check=True,
26                 shell=True,
27             )
28             print("Osdag was installed successfully.")
29             success = True # Exit the loop if the command succeeds
30
31         except subprocess.CalledProcessError as e:
32             print(f"Attempt {attempt} failed. Retrying... Error: {e}")
33             attempt += 1
34             time.sleep(10) # Wait 10 seconds before retrying
35
36         except FileNotFoundError as e:
37             print(f"Error: {e}")
38             break # Exit the loop if Conda is not installed and cannot
39                 proceed
40
41     if attempt == 10 and not success:
42         print("Failed to install Osdag after multiple attempts. Please
43             check your network or Conda setup.")
44
45 def main():
46     try:
47         install_miniconda()
48         #install_tinytex()
49         create_conda_env()
50         print("Installation completed successfully. Ready to use.")

```

```
48     except Exception as e:
49         print(f"Installation failed: {e}")
50
51 if __name__ == "__main__":
52     #makes sure code runs directly from this script and not some external
53     func
54     main()
55 %----- end code -----
```

# Chapter 3

## Internship Task 1

## Osdag LaTeX Environment

### 3.1 Task 1: Problem Statement

Creating Osdag's own Tex Environment such that Osdag is self-reliant for compiling Tex reports and generating .pdf's and now is not reliant on a Tex distribution installation like MikTex or TinyTex.

### 3.2 Task 1: Tasks Done

Osdag achieves its own LaTeX Environment through a texmf-tree structure. texmf stands for Tex Metafont. A texmf-tree is a specific directory structure that LaTeX understands when locating packages.

This is the default state of a texmf tree in a Tex installation.

```
bin
├── windows.
│   ├── pdflatex.exe.
│   └── other essential binaries
├── texmf-config
│   ├── .configuration files - specific to user
│   └── ls-R
├── texmf-dist
│   ├── bibtex
│   ├── dvipdfmx
│   ├── dvips
│   ├── fonts
│   └── makeindex
```

```
|
|_ metafont
|_ mft
|_ scripts
|_ tex
|_ texconfig
|_ web2c
|_ texmf-local
|_ texmf-var
```

### 3.3 Task 1: Significance of each branch of the texmf tree

#### 1. bin/

Contains the binaries (executables) for TeX tools like pdflatex, xelatex, lualatex, kpsewhich etc. and also the .dll files for working with Tex on a Win machine.

When running TeX commands from the terminal, the executables in this folder are used.

#### 2. texmf-config/

Stores configuration files for the TeX system.

Used for user-specific settings, such as package configurations and font mappings. texmf-config/ is where TeX system settings can be customized without modifying texmf-dist/. (if we ever want to include our custom font mappings)

#### 3. texmf-dist/

Contains the default installed TeX packages (such as article.cls, beamer.cls, .sty files).

It is the main package repository of the TinyTeX installation.

#### 4. texmf-local/

A directory meant for locally installed packages and custom TeX files.

This is where maintainer can install additional .sty files that persist even after updating TinyTeX.

Say we want a specific .sty package that we want to freeze at a specific version, that a future update does not tamper with, this is where it'll be.

#### 5. texmf-var/

Stores generated files like font caches and auxiliary data produced when compiling .tex documents.

Consists of format files, (.fmt) that help speed up document compilation by avoiding the need to regenerate macro definitions every time.

## 3.4 Task 1: Python Code

The Python Code designed for this task was added to the files - reportGenerator\_latex.py and SectionModeller\_Latex.py.

### 3.4.1 Description of the Script

The script has two important parts:

- **\*\*Initialize Input Variables\*\***: The additional code starts by initializing environment variables that pylatex uses to identify where the Tex packages are located.
- **\*\*Compiler Declaration\*\***: The location to pdflatex.exe engine, used to compile .pdf reports from .tex source files is declared in the compiler flag of the generate\_pdf() command from pylatex.

### 3.4.2 Python Code

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

Listing 3.1: Initializing Environment Variables for texmf

```
1 %-----begin code-----
2
3 os.environ['TEXMFHOME'] = os.path.abspath("data/ResourceFiles/osdag-
   latex-env/texmf-dist")
4 sty_pkgs = str(files("osdag.data.ResourceFiles.osdag-latex-env.texmf-
   dist")).replace("\\", "/")
5 pkg_resources = [f'{sty_pkgs}/amsmath', f'{sty_pkgs}/graphics', f'{
   sty_pkgs}/needspace']
6 texinp = os.environ.get('TEXINPUTS', ' ')
7
8 pkg_path = ";".join(pkg_resources)
9 os.environ['TEXINPUTS'] = f'{pkg_path};{texinp}'
10 %-----end code -----
```

### 3.4.3 Explanation of the Code in Initializing Environment Variables Snippet

- **\*\*Line 3\*\***: Sets the TEXMFHOME environment variable to point to the absolute path of a custom LaTeX environment.
- **\*\*Line 4\*\***: Resolves the filesystem path of the directory containing LaTeX style files using `importlib.resources.files()`.
- **\*\*Line 5\*\***: Prepares a list of paths to specific LaTeX package directories (`amsmath`, `graphics`, `needspace`) inside the environment.
- **\*\*Line 6\*\***: Retrieves the current value of the TEXINPUTS environment variable, if set, or defaults to a single space ' '.
- **\*\*Line 8\*\***: Joins the paths in `pkg_resources` into a single string using `;` as a separator.
- **\*\*Line 9\*\***: Sets the updated TEXINPUTS environment variable to include the custom package directories plus any previously set paths.

Listing 3.2: Declaring Compiler for Pylatex in `SectionModeller_Latex.py`

```
1 %-----begin code-----
2
3 latex_executable = os.path.abspath("data/ResourceFiles/osdag-latex-env/
  bin/windows/pdflatex.exe")
4 doc.generate_pdf(filename, compiler=latex_executable, clean_tex=False)
5
6 %-----end code -----
```

### 3.4.4 Explanation of the Code Snippet added to `SectionModeller_Latex.py`

- `os.path.abspath(...)`:  
Converts the relative path to the `pdflatex.exe` executable into an absolute path, assuming the path is correct relative to the current working directory.

- `doc.generate_pdf(...)`:

This uses the `pdflatex.exe` located at that absolute path to compile the LaTeX document.

The `filename` flag is used to set the name of the `.tex` file.

`compiler=latex_executable`: Specifies which LaTeX compiler to use.

`clean_tex=False`: Keeps the `.tex` file after compilation instead of deleting it.

Listing 3.3: Declaring Compiler for Pylatex in `reportGenerator_latex.py`

```

1  %-----begin code-----
2
3  script_dir = os.path.dirname(os.path.abspath(__file__))
4
5  # Go one level up to reach the parent directory
6  parent_dir = os.path.dirname(script_dir)
7
8  # Construct the path to pdflatex.exe
9  latex_executable = os.path.join(parent_dir, "data", "ResourceFiles", "
    osdag-latex-env", "bin", "windows", "pdflatex.exe")
10
11 # Ensure the path is absolute
12 latex_executable = os.path.abspath(latex_executable)
13 doc.generate_pdf(filename, compiler=latex_executable, clean_tex = False
    )
14
15 %-----end code -----

```

### 3.4.5 Explanation of the Code Snippet added to `reportGenerator_latex.py`

- `script_dir`

Gets the absolute path to the current script file (`__file__` is the path of the currently running Python script).

- `parent_dir`

Moves one level up in the directory hierarchy (e.g., if the script is in `.../scripts`, this goes to `.../`).



- `os.path.join(...)`

Builds the full path to `pdflatex.exe` from the parent directory, ensuring that even if the script is run from a different directory, the compiler path is correctly located.

- `os.path.abspath(...)`

Ensures the path is absolute (technically redundant here, but good practice).

- `doc.generate_pdf(...)`

Same as in snippet 1: compiles the `.tex` file using the specified LaTeX compiler and retains the `.tex` file.

# Chapter 4

## Internship Task 2

### Tex Manager -Maintainer's Workflow

#### 4.1 Task 2 Problem Statement

The task was to create a Maintainer's Workflow for managing updates to Tex packages using Tex Manager, Git and Git Submodules.

#### 4.2 Task 2: Workflow

Say an update rolls out for a .sty package that we have in the 177 MB of files (osdag-latex-env), referenced as a submodule on a separate git repo.

To check for updates the maintainer, on their local machine with a Tex distro installed, types in a command on cmd:

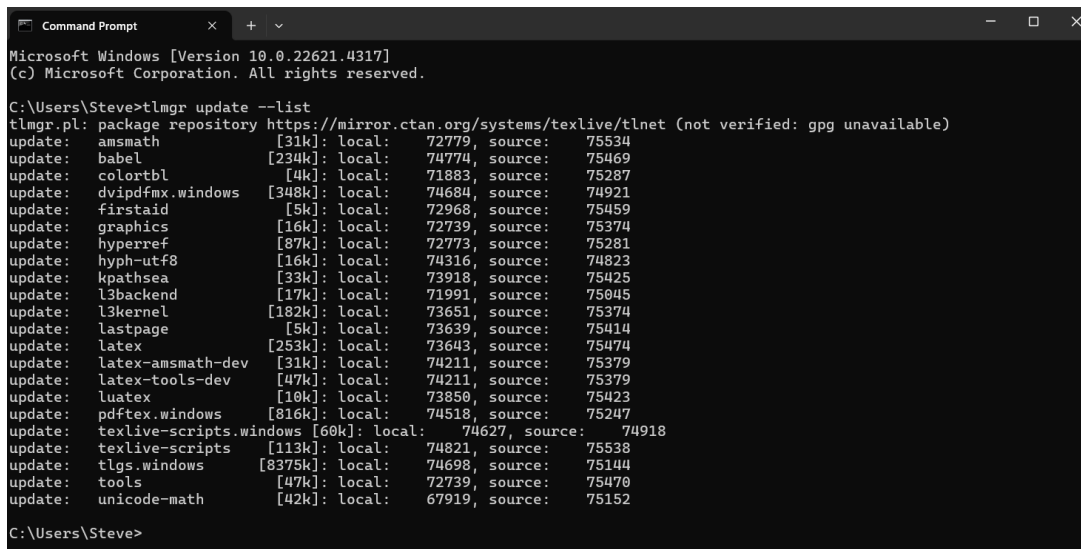
Depends the Tex distro the maintainer is using, but for TinyTex:

**tlmgr update --list**

This will show which .sty packages have updates available.

**tlmgr update --all** will update all packages to the latest version.

TinyTex by itself is in the form of a texmf tree. So the packages can be updated by using the package manager(tlmgr) for the maintainer, and maintainer can then decide which package updates or specific versions need to be pushed to the external repo (which



```
Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Steve>tlmgr update --list
tlmgr.pl: package repository https://mirror.ctan.org/systems/texlive/tlnet (not verified: gpg unavailable)
update: amsmath [31k]: local: 72779, source: 75534
update: babel [234k]: local: 74774, source: 75469
update: colortbl [4k]: local: 71883, source: 75287
update: dvipdfmx.windows [348k]: local: 74684, source: 74921
update: firstaid [5k]: local: 72968, source: 75459
update: graphics [16k]: local: 72739, source: 75374
update: hyperref [87k]: local: 72773, source: 75281
update: hyph-utf8 [16k]: local: 74316, source: 74823
update: kpathsea [33k]: local: 73918, source: 75425
update: l3backend [17k]: local: 71991, source: 75045
update: l3kernel [182k]: local: 73651, source: 75374
update: lastpage [5k]: local: 73639, source: 75414
update: latex [253k]: local: 73643, source: 75474
update: latex-amsmath-dev [31k]: local: 74211, source: 75379
update: latex-tools-dev [47k]: local: 74211, source: 75379
update: luatex [10k]: local: 73850, source: 75423
update: pdftex.windows [816k]: local: 74518, source: 75247
update: texlive-scripts.windows [60k]: local: 74627, source: 74918
update: texlive-scripts [113k]: local: 74821, source: 75538
update: tlgs.windows [8375k]: local: 74698, source: 75144
update: tools [47k]: local: 72739, source: 75470
update: unicode-math [42k]: local: 67919, source: 75152

C:\Users\Steve>
```

Figure 4.1: tlmgr update --list

is referenced as a submodule by the main Osdag repo).

tlpkg is the folder which contains the Tex Live package manager(tlmgr) and other dependencies used by TinyTex to keep its packages updated. Tlpkg is retained for the maintainer and removed for the Osdag user

The maintainer decides which packages get updates by using a .gitignore file to mention the files that are not tracked by git or simply put - don't get updates.

By using .gitignore, in the maintainer's TinyTex installed directory, tlpkg folder is ignored by git while tracking files.

Effectively we're adding the external repo as a remote to the local directory of TinyTex of the maintainer, using .gitignore to untrack the files we don't need to be pushing to remote repo, updating the packages using tlmgr for the maintainer and then pushing the updates to remote repo using git.

This removes the step in the initial approach where the maintainer had to first note which packages need to be updated, and then go off to CTAN to manually download the updated packages then update the external repo manually by adding all the updated packages and committing them.

**Instead let tlmgr and git do the job.**

## 4.3 Task 2: Setting Up Environment for the Maintainer

(these steps are required to be executed only once i.e only for the initial setup for the maintainer)

1.Now since the maintainer has a Tex distro installed, within the directory where TinyTex is installed for the maintainer, initialize a git repo.

In general cases, TinyTex will be installed for the maintainer on E.g

C:\Users\UserName\AppData\Roaming\TinyTex

Within TinyTex folder, initialize git.

```
>>git init
```

```
>>touch .gitignore
```

Figure 4.2 shows how the default directory structure of a TinyTex installation looks like: i.e the default structure of a texmf-tree

Name	Date modified	Type	Size
.git	11-04-2025 16:32	File folder	
bin	03-04-2025 22:30	File folder	
temp	03-06-2025 18:00	File folder	
texmf-config	03-04-2025 22:30	File folder	
texmf-dist	10-06-2025 12:03	File folder	
texmf-local	03-04-2025 22:30	File folder	
texmf-var	03-04-2025 22:30	File folder	
tlpkg	10-06-2025 12:03	File folder	
.gitignore	11-04-2025 16:23	Git Ignore Source ...	1 KB
.tinytex	13-03-2025 14:07	TINYTEX File	1 KB
install-tl	03-04-2025 23:28	File	124 KB
install-tl-windows	03-04-2025 23:26	Windows Batch File	6 KB
LICENSE.CTAN	03-06-2025 18:00	CTAN File	3 KB
LICENSE.TL	03-06-2025 18:00	TL File	6 KB
release-texlive	03-06-2025 18:00	Text Document	1 KB
texmf.cnf	13-03-2025 14:07	CNF File	1 KB
texmfcnf	13-03-2025 14:07	Lua Source File	1 KB
tl-tray-menu	03-04-2025 23:26	Application	49 KB

Figure 4.2: Default structure of the texmf tree

2. touch .gitignore creates a .gitignore file. Navigate to the TinyTeX folder and open .gitignore in any editor.

Add all files and folders except:

- bin
- texmf-config
- texmf-dist
- texmf-local
- texmf-var

A screenshot of a code editor showing the contents of a .gitignore file. The file path is displayed at the top: C:\> Users > Steve > AppData > Roaming > TinyTeX > .gitignore. The file contains 12 lines of text, each preceded by a line number from 1 to 12. The lines are: 1 temp/, 2 tlpkg/, 3 .tinytex, 4 install-tl, 5 install-tl-windows.bat, 6 LICENSE.CTAN, 7 LICENSE.TL, 8 release-texlive.txt, 9 texmf.cnf, 10 texmf.cnf.lua, 11 tl-tray-menu.exe, and 12 (empty line).

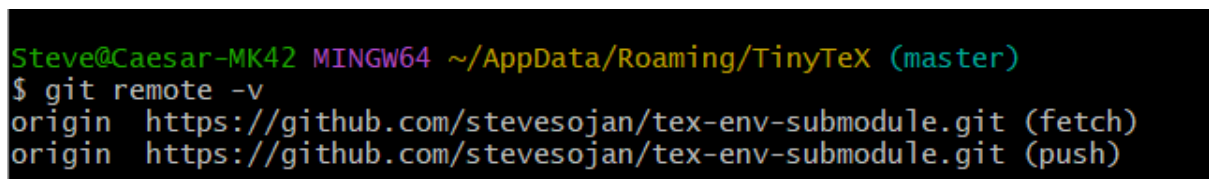
```
C: > Users > Steve > AppData > Roaming > TinyTeX > .gitignore
1  temp/
2  tlpkg/
3  .tinytex
4  install-tl
5  install-tl-windows.bat
6  LICENSE.CTAN
7  LICENSE.TL
8  release-texlive.txt
9  texmf.cnf
10 texmf.cnf.lua
11 tl-tray-menu.exe
12
```

Figure 4.3: inside .gitignore for the maintainer

into .gitignore. Only the files and folders other than the above mentioned have to be added to .gitignore.

Open .gitignore and add everything except the folders mentioned above on .gitignore.

Now here I've used my Github Repo named 'tex-env-submodule' as an external repo. The external repo must be added as a remote to the user's local repo.

A screenshot of a terminal window showing the output of the 'git remote -v' command. The prompt is 'Steve@Caesar-MK42 MINGW64 ~/AppData/Roaming/TinyTeX (master)'. The output shows two entries for the 'origin' remote, both pointing to 'https://github.com/stevesojan/tex-env-submodule.git', with '(fetch)' and '(push)' respectively.

```
Steve@Caesar-MK42 MINGW64 ~/AppData/Roaming/TinyTeX (master)
$ git remote -v
origin  https://github.com/stevesojan/tex-env-submodule.git (fetch)
origin  https://github.com/stevesojan/tex-env-submodule.git (push)
```

Figure 4.4: Add the external repo (here tex-env-submodule) to this local repo as a remote.

Link to external repo: <https://github.com/stevesojan/tex-env-submodule>

Now our external repo is populated with the texmf tree as per its original structure. The aim is that this texmf tree is supplied to Osdag's Tex environment, through an external repo which is referenced as a submodule to the main repo (osdag-admin/Osdag).

## 4.4 Task 2: Implementing Git Submodules

Git Submodules is implemented such that updates to Osdag's Tex Environment can be handled in a separate repository but the required state of the texmf-tree can be referenced by the main repository.

I cloned Osdag's repository(osdag-admin/Osdag) and referenced tex-env-submodule as a submodule to osdag-admin/Osdag within the src/osdag/data/ResourceFiles\osdag-latex-env folder.

The commit hash of the submodule is implicitly stored in the **main repo's commit history**, and git submodule update reads from that.

Navigate into your main repo (osdag-admin/Osdag):

```
>>cd path/to/Osdag
```

Add the submodule at the desired path:

**Syntax:**

```
>>git submodule add <repository-url> [<path>]
```

```
>>git submodule add https://github.com/your-username/tex-env-submodule.git  
src/osdag/data/ResourceFiles/osdag-latex-env
```

Replace `https://github.com/your-username/tex-env-submodule.git` with your actual submodule URL.

This will initialize a .gitmodules folder in the main repo.

Make sure in the .gitmodules, the path looks something like in Figure 4.5.

Where the url is replaced with the link of the external repo, that you would want to reference as the submodule.

The path variable is the path to where the submodule needs to be added with respect to the root of the main repo.

Commit the changes to osdag-admin:

```

C: > Users > Steve > OneDrive > Documents > osdag-admin\Osdag forked > Osdag > .gitmodules
1  [submodule "src/osdag/data/ResourceFiles/osdag-latex-env"]
2      path = src/osdag/data/ResourceFiles/osdag-latex-env
3      url = https://github.com/stevesojan/tex-env-submodule.git
4

```

Figure 4.5: Inside .gitmodules in main repo

```

>>git add .gitmodules src/osdag/data/ResourceFiles/osdag-latex-env
>>git commit -m "Added tex-env-submodule as a submodule"

```

Say if the packages get updated and the new updates are pushed to the external repo. Now all we need to do is change the commit hash that is currently being referenced by osdag-admin/Osdag.git

Now we will look at available commits and hash references.

## 4.5 Task 2: To view all available commits in the Submodule

First, **cd** into your submodule directory:

```
>>cd src/Osdag/data/ResourceFiles/osdag-latex-env
```

Then run:

```
>>git log --oneline
```

This will list all the commits in the submodule's history.

```

The hash that has HEAD -> mentioned is
the exact commit the main repository is referencing.

```

## 4.6 Task 2: To change the commit hash referenced

(navigate to the submodule referenced folder i.e osdag-latex-env, on git bash)

```

>>cd src/Osdag/data/ResourceFiles/osdag-latex-env
>>git checkout <desired_commit_hash>

```

```

Steve@Caesar-MK42 MINGW64 ~/OneDrive/Documents/osdag-admin
ster)
$ cd src/Osdag/data/ResourceFiles/osdag-latex-env

Steve@Caesar-MK42 MINGW64 ~/OneDrive/Documents/osdag-admin
/Osdag/data/ResourceFiles/osdag-latex-env ((70547be...))
$

Steve@Caesar-MK42 MINGW64 ~/OneDrive/Documents/osdag-admin
/Osdag/data/ResourceFiles/osdag-latex-env ((70547be...))
$ git log --oneline
70547be (HEAD, origin/master, origin/HEAD, master) update
e1f26f3 updated gitignore
b62f15c initial commit

```

Figure 4.6: To view the commits made in a submodule

Then return to your main repo root and update the pointer:

Main-repo-root is the folder where Osdag.git is located for the maintainer locally on their PC.

```

>>cd <main-repo-root>
>>git add src/osdag/data/ResourceFiles/osdag-latex-env
>>git commit -m "submodule update to another commit"
>>git push

```

```

Steve@Caesar-MK42 MINGW64 ~/OneDrive/Documents/osdag-adminOsdag forked/Osdag/src/osdag/data/ResourceFiles/osdag-latex-env (master)
$ git checkout e1f26f3
Note: switching to 'e1f26f3'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at e1f26f3 updated gitignore
Steve@Caesar-MK42 MINGW64 ~/OneDrive/Documents/osdag-adminOsdag forked/Osdag/src/osdag/data/ResourceFiles/osdag-latex-env ((e1f26f3...))
$ cd ..
Steve@Caesar-MK42 MINGW64 ~/OneDrive/Documents/osdag-adminOsdag forked/Osdag/src/osdag/data/ResourceFiles (master)
$ cd ..
Steve@Caesar-MK42 MINGW64 ~/OneDrive/Documents/osdag-adminOsdag forked/Osdag/src/osdag/data (master)
$ cd ..
Steve@Caesar-MK42 MINGW64 ~/OneDrive/Documents/osdag-adminOsdag forked/Osdag/src/osdag (master)
$ cd ..
Steve@Caesar-MK42 MINGW64 ~/OneDrive/Documents/osdag-adminOsdag forked/Osdag/src (master)
$ cd ..
Steve@Caesar-MK42 MINGW64 ~/OneDrive/Documents/osdag-adminOsdag forked/Osdag (master)
$ git add src/osdag/data/ResourceFiles/osdag-latex-env
Steve@Caesar-MK42 MINGW64 ~/OneDrive/Documents/osdag-adminOsdag forked/Osdag (master)
$ git commit -m 'submodule updated to another commit'
[master 76c19a1] submodule updated to another commit
1 file changed, 1 insertion(+), 1 deletion(-)

```

Figure 4.7: Reference hash changed

Here, I changed the commit hash reference to e1f26f3. By following the same steps, the maintainer can change the commit hash reference to any previous version of the packages i.e a rollback, in an instant. Say the updated Tex packages are not correctly compiling the .tex file - the maintainer can change the commit reference to a previous ver-



sion and next time Osdag is updated, the right versions of the packages will be reinstated.

Here is a flowchart of the entire workflow. In the flowchart the green blocks are the steps to execute, the grey blocks are purely comments for the preceding green block.

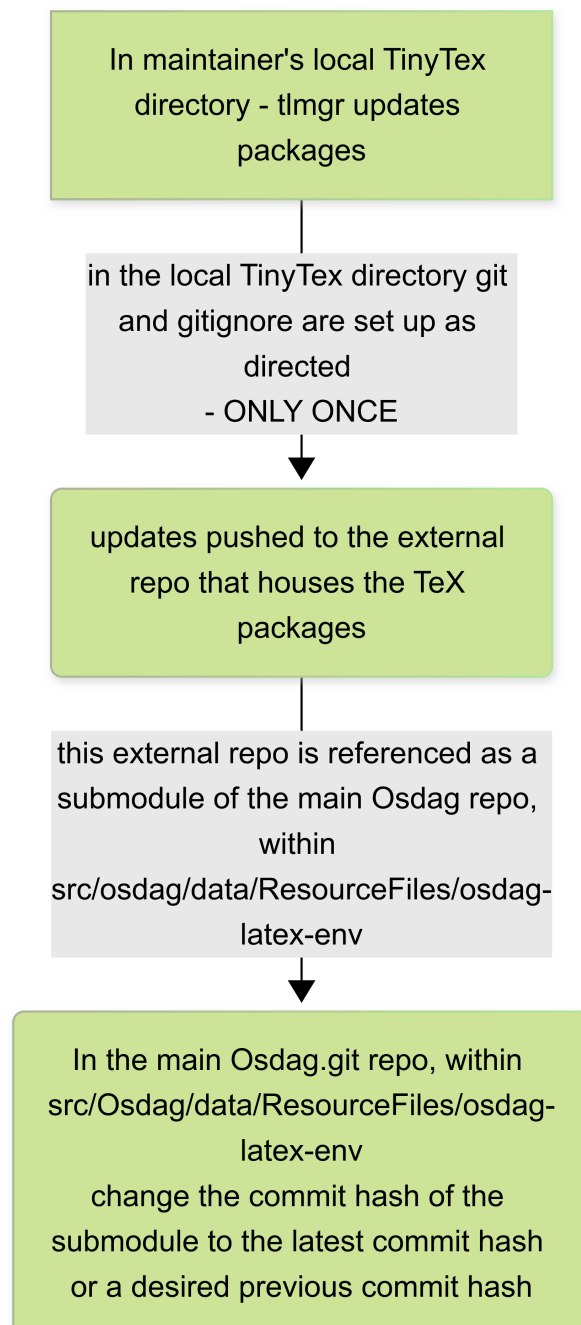


Figure 4.8: Tex Maintainer's Workflow

# Chapter 5

## Internship Task 3

## LaTeX Testing Suite

### 5.1 Task 3 Problem Statement

The task was to develop a testing suite for LaTeX that performs all necessary checks to ensure that the dependencies and the entire report generation system are functioning correctly and remain intact.

### 5.2 Task 3: Arrange, Act and Assert Approach

The testing suite is configured in the arrange, act and assert pattern:

- Arrange is used to set the dependencies, define helper functions and setting up environment paths. In short, setting up any dependencies that we may have for the actual tests
- Act is the test itself, the core logic as to how it tests for a certain criteria using helper functions and other dependencies.
- Assert is used to check the final decision logic that ‘Act’ has arrived at - whether a test is Pass or Fail.

## 5.3 Task 3: Checks Done

The script for LaTeX Testing Suite is `test_tex_diagnostics.py`. Ideally it must be located in:

```
\osdag\data\test_tex_diagnostics.py
```

The script is built to test if files are available in `osdag-latex-env`. So it must be located in `data` folder (within which `osdag-latex-env` is located).

- Checks if `pdflatex` exists either globally through `cmd` path (in the case that a user already has an existing Tex installation) or in `osdag-latex-env`
- Checks if `Latexmk` is available in `osdag-latex-env`
- Checks if the essential `.sty` files are present. (also includes all `.sty` files used by any `.tex` file generated by Osdag currently for any module)
- `mports` and checks functioning of the very essential `.sty` packages listed in `REQUIRED STY FILES` used by Osdag. Sets environment variables to help LaTeX find custom packages.
- The final Osdag Diagnostics Report Compilation.

## 5.4 Task 3: Helper Functions

The current testing suite for LaTeX uses 3 helper functions, which are used by the main testing functions. They come under the Arrange bracket of the Arrange, Act and Assert pattern.

They are defined at the top of the script in this order:

### 1. `is_pdflatex_in_path()`

Purpose: Checks if `pdflatex` is available in the system's environment `PATH`.

Returns: True if found (i.e., system-wide installation exists or in `data/ResourceFiles/osdag-latex-env/bin/windows/pdflatex.exe`), else False.

Listing 5.1: `is_pdflatex_in_path()`

```
1 %-----begin code-----
```

```

2 def is_pdflatex_in_path():
3     return shutil.which("pdflatex") is not None
4 %----- end code -----

```

## 2. is\_pdflatex\_in\_osdag\_env()

Purpose: Checks if a fallback pdflatex.exe exists inside the Osdag-provided LaTeX environment directory.

Returns: True if pdflatex.exe is present in osdag-latex-env/bin/windows, else False.

Listing 5.2: is\_pdflatex\_in\_osdag\_env()

```

1 %-----begin code-----
2 return os.path.exists(FALLBACK_PDFLATEX_PATH)
3 %----- end code -----

```

## 3. find\_sty\_file(root\_dir, target\_file)

Purpose: Recursively searches for a .sty file (LaTeX style file) named target\_file inside root\_dir.

Returns: The full path to the .sty file if found, otherwise None.

Listing 5.3: find\_sty\_file(root\_dir,target\_file)

```

1 %-----begin code-----
2 def find_sty_file(root_dir, target_file):
3     for dirpath, _, filenames in os.walk(root_dir):
4         if target_file in filenames:
5             return os.path.join(dirpath, target_file)
6     return None
7 %----- end code -----

```

# 5.5 Task 3: Tests Conducted

**Test 1:** test\_pdflatex\_available() Checks whether the LaTeX compiler pdflatex is available either: globally in the system's PATH, or locally in the osdag-latex-env.

Pass Criteria: pdflatex must be found in at least one location.

Listing 5.4: test\_pdflatex\_available()

```

1 %-----begin code-----
2 def test_pdflatex_available():

```

```

3     in_path = is_pdflatex_in_path()
4     in_env = is_pdflatex_in_osdag_env()
5     available = in_path or in_env
6     diagnostics_log.append(f"Test {test_counter[0]} - pdflatex
    availability: {'PASSED' if available else 'FAILED'}")
7     test_counter[0] += 1
8     assert available, "pdflatex is not installed globally or in osdag-
    latex-env"
9 %----- end code -----

```

**Test 2:** test\_latexmk\_present() Checks if the latexmk.exe executable exists inside the osdag-latex-env path.

Pass Criteria: The file latexmk.exe should be present in LATEX\_ENV\_PATH/bin/windows.

Listing 5.5: test\_latexmk\_present()

```

1 %-----begin code-----
2 def test_latexmk_present():
3     latexmk_path = os.path.join(LATEX_ENV_PATH, 'bin', 'windows', '
    latexmk.exe')
4     exists = os.path.exists(latexmk_path)
5     diagnostics_log.append(f"Test {test_counter[0]} - latexmk presence:
    {'PASSED' if exists else 'FAILED'}")
6     test_counter[0] += 1
7     assert exists, "latexmk executable not found in osdag-latex-env"
8 %----- end code -----

```

**Test 3:** test\_required\_sty\_files\_present() Runs once for each .sty file in REQUIRED\_STY\_FILES using pytest.mark.parametrize. It searches recursively in LATEX\_ENV\_PATH for each .sty file.

Pass Criteria: Each .sty file must be found in the directory structure.

Listing 5.6: test\_required\_sty\_files\_present(sty\_file)

```

1 %-----begin code-----
2 def test_required_sty_files_present(sty_file):
3     found = find_sty_file(LATEX_ENV_PATH, sty_file) is not None
4     diagnostics_log.append(f"Test {test_counter[0]} - {sty_file}: {'
    PASSED' if found else 'FAILED'}")
5     test_counter[0] += 1
6     assert found, f"{sty_file} not found in LaTeX environment"

```

```
7 %----- end code -----
```

**Test 4:** `test_compile_latex_with_pylatex()` Imports and checks functioning of the very essential .sty packages listed in `REQUIRED_STY_FILES` used by Osdag. Sets environment variables to help LaTeX find custom packages.

Generates a.pdf from the packages it uses by the name of 'test4\_latex\_testing\_suite'.

Compiles it using PyLaTeX.

Pass Criteria: The document compiles without any LaTeX CompilerError.

Listing 5.7: `test_compile_latex_with_pylatex()`

```
1 %-----begin code-----
2 def test_compile_latex_with_pylatex():
3
4     os.environ['TEXMFHOME'] = os.path.abspath("data/ResourceFiles/osdag
5         -latex-env/texmf-dist")
6
7     os.environ["TEXINPUTS"] = os.path.abspath("data/ResourceFiles/osdag
8         -latex-env/texmf-dist") + os.pathsep + os.environ.get("TEXINPUTS
9         ", "")
10
11     sty_pkgs = os.path.abspath(os.path.join("data", "ResourceFiles", "
12         osdag-latex-env", "texmf-dist"))
13
14     sty_pkgs = sty_pkgs.replace("\\", "/") # Normalize path for LaTeX
15         on Windows
16
17     pkg_resources = [f'{sty_pkgs}/amsmath', f'{sty_pkgs}/graphics', f'{
18         sty_pkgs}/needspace']
19
20     texinp = os.environ.get('TEXINPUTS', ' ')
21
22     pkg_path = ";".join(pkg_resources)
23
24     os.environ['TEXINPUTS'] = f'{pkg_path};{texinp}'
25
26     geometry_options = {
27         "a4paper": True,
28         "top": "2cm",
29         "bottom": "2cm",
30         "left": "2.5cm",
31         "right": "2.5cm"
32     }
33
34     doc = Document(documentclass="article", geometry_options=
```

```

        geometry_options)
23
24     # Import all packages
25     doc.packages.append(Package('inputenc', options='utf8'))
26     doc.packages.append(Package('fontenc', options='T1'))
27     doc.packages.append(Package('lmodern'))
28     doc.packages.append(Package('amsmath'))
29     doc.packages.append(Package('graphicx'))
30     doc.packages.append(Package('xcolor'))
31     doc.packages.append(Package('color'))
32     doc.packages.append(Package('fancyhdr'))
33     doc.packages.append(Package('geometry'))
34     doc.packages.append(Package('lastpage'))
35     doc.packages.append(Package('multirow'))
36     doc.packages.append(Package('colortbl'))
37     doc.packages.append(Package('array'))
38     doc.packages.append(Package('longtable'))
39     doc.packages.append(Package('tabularx'))
40     doc.packages.append(Package('needspace'))
41     doc.packages.append(Package('parskip'))
42     doc.packages.append(Package('ltxcmds'))
43     doc.packages.append(Package('kvoptions'))
44     doc.packages.append(Package('kvsetkeys'))
45
46     doc.preamble.append(NoEscape(r'''
47         \usepackage{fancyhdr}
48         \pagestyle{fancy}
49         \renewcommand{\headrulewidth}{0pt}
50
51         \fancyhead[L]{
52             \begin{minipage}[t]{0.6\textwidth}
53                 \vspace{-1em}
54                 {\Large\bfseries Osdag Diagnostics Report - Test 4}
55                 \\\[0.5em]
56                 {\small \today}
57             \end{minipage}
58
59         '''))

```

```

60
61
62
63     doc.append('This document confirms the successful import and basic
        usage of essential .sty files used by Osdag.\n')
64
65     with doc.create(Subsection('amsmath')):
66         doc.append(NoEscape(r'$\int_0^\infty e^{-x^2} dx = \frac{\sqrt{
            \pi}}{2}$'))
67
68     with doc.create(Subsection('xcolor & color')):
69         doc.append(NoEscape(r'\definecolor{OsdagGreen}{RGB}{153,169,36}
            '))
70         doc.append(NoEscape(r'\textcolor{OsdagGreen}{This is Osdag
            Green Colour}'))
71
72     with doc.create(Subsection('multirow, colortbl, array')):
73         doc.append(NoEscape(r'''
74 \begin{tabular}{|c|c|}
75 \hline
76 \multirow{2}{*}{A} & Row 1 \\
77                    & Row 2 \\
78 \hline
79 \end{tabular}
80 '''))
81
82     with doc.create(Subsection('longtable, tabularx')):
83         doc.append(NoEscape(r'''
84 \rowcolors{2}{gray!10}{white}
85 \begin{longtable}{|>\raggedright\arraybackslash p{4cm}|p{9cm}|}
86 \hline
87 Feature & Description \\
88 \hline
89 Color Alternating & Tested with colortbl and xcolor \\
90 Custom Column Align & Tested with array \\
91 Auto Page-Break Table & longtable allows breaking \\
92 \hline
93 \end{longtable}
94 '''))

```



```

95
96     # Export
97     doc.generate_pdf('test4_latex_testing_suite', clean=True, compiler=
        PDFLATEX_PATH, clean_tex=False)
98     diagnostics_log.append(f"Test {test_counter[0]} - LaTeX compilation
        with all .sty files: PASSED")
99     test_counter[0] += 1
100
101
102 %----- end code -----

```

**Test 5:** `test_compile_latex_with_pylatex_report()` The final diagnostics report compilation. The final report named “osdag\_diagnostics\_report.pdf” is stored in the same folder from wherever the script is run.

This includes: Setting up the page geometry and headers.

Drawing a metadata table with system info (hostname, user, etc.).

Listing all test results.

Pass Criteria: If the report compiles and displays the result table, the test passes.

Listing 5.8: `test_compile_latex_with_pylatex_report()`

```

1 %-----begin code-----
2 def test_compile_latex_with_pylatex_report():
3
4     # Set LaTeX environment paths
5     os.environ['TEXMFHOME'] = os.path.abspath("data/ResourceFiles/osdag
        -latex-env/texmf-dist")
6     os.environ["TEXINPUTS"] = os.path.abspath("data/ResourceFiles/osdag
        -latex-env/texmf-dist") + os.pathsep + os.environ.get("TEXINPUTS
        ", "")
7
8     sty_pkgs = os.path.abspath(os.path.join("data", "ResourceFiles", "
        osdag-latex-env", "texmf-dist"))
9     sty_pkgs = sty_pkgs.replace("\\", "/") # Normalize path for LaTeX
        on Windows
10    pkg_resources = [f'{sty_pkgs}/amsmath', f'{sty_pkgs}/graphics', f'{
        sty_pkgs}/needspace']
11    texinp = os.environ.get('TEXINPUTS', ' ')
12    pkg_path = " ".join(pkg_resources)

```

```

13 os.environ['TEXINPUTS'] = f'{pkg_path};{texinp}'
14 base_dir = os.path.dirname(os.path.abspath(__file__))
15 pkg_images = os.path.join(base_dir, "data", "ResourceFiles", "
    images")
16 geometry_options = {
17     "a4paper": True,
18     "top": "4cm",
19     "hmargin": "2cm",
20     "headheight": "100pt",
21     "footskip": "100pt",
22     "bottom": "2.5cm"
23 }
24 imgpath_osdagheader = os.path.join(pkg_images, "Osdag_header_report
    .png").replace("\\", "/")
25 doc = Document(documentclass="article", geometry_options=
    geometry_options)
26
27 doc.packages.append(Package('inputenc', options='utf8'))
28 doc.packages.append(Package('graphicx'))
29 doc.packages.append(Package('lmodern'))
30 doc.preamble.append(NoEscape(r'\renewcommand{\familydefault}{\
    sfdefault}'))
31 doc.packages.append(Package('helvet'))
32 doc.packages.append(Package('fancyhdr'))
33 doc.packages.append(Package('xcolor'))
34 doc.packages.append(Package('array'))
35 doc.packages.append(Package('xcolor', options='table'))
36
37 # Add Osdag green color
38 doc.preamble.append(NoEscape(r'\definecolor{OsdagGreen}{RGB
    }{153,169,36}'))
39
40 # Add Osdag header image
41 doc.preamble.append(NoEscape(r'''
42     \usepackage{fancyhdr}
43     \pagestyle{fancy}
44     \renewcommand{\headrulewidth}{1pt}
45
46     \fancyhead[L]{

```

```

47         \begin{minipage}[t]{0.6\textwidth}
48             \vspace{-3em}
49             {\Large\bfseries Osdag Diagnostics Report} \\[0.5em]
50             {\small \today}
51         \end{minipage}
52     }
53
54     \fancyhead[R]{
55         \begin{minipage}[t]{0.3\textwidth}
56     '''))
57
58     # Insert the image with StandAloneGraphic
59     doc.preamble.append(StandAloneGraphic(image_options="width=4.40cm,
60         height=1.1cm",filename=imgpath_osdagheader))
61
62     doc.preamble.append(NoEscape(r'''
63         \end{minipage}
64     '''))
65     current_time = datetime.now().strftime("%d-%m-%Y | %H:%M:%S")
66     # Add metadata table
67     with doc.create(Tabular(r'|>{\raggedright\arraybackslash}p{5cm
68         }||>{\raggedright\arraybackslash}p{10cm}|')) as table:
69
70         table.add_hline()
71         table.add_row(('Test', 'LaTeX Testing Suite'), color='
72             OsdagGreen')
73         table.add_hline()
74         table.add_row(('Date & Time', current_time), color='OsdagGreen'
75             )
76         table.add_hline()
77         table.add_row(('Computer Name', f'{socket.gethostname()}'),
78             color='OsdagGreen')
79         table.add_hline()
80         table.add_row(('User', f'{os.getlogin()}'), color='OsdagGreen')
81         table.add_hline()
82
83     # Description section
84     with doc.create(Section("Checks Done:", numbering=False)):

```

```

81     doc.append(NoEscape(r'''
82         The current LaTeX testing suite checks for the presence of
83         the following dependencies:
84         \begin{itemize}
85             \item Presence of pdflatex.exe either in system path or
86                 present in osdag\_latex\_env/bin/windows
87             \item Presence of latexmk.exe
88             \item Presence of the essential .sty files. Also
89                 includes all .sty files used by any .tex file
90                 generated by Osdag currently for any module.
91             \item Calls each of the essential .sty files and
92                 compiles a sample report where each package is used.
93                 If compilation is successful it deletes the .tex
94                 file and resultant .pdf, .aux, .log. If compilation
95                 is unsuccessful, the test fails i.e either a .sty
96                 package is corrupt or it is missing.
97         \end{itemize}
98     '''))
99
100 diagnostics_log.append(f"Test {test_counter[0]} - Osdag Diagnostics
101     Report compilation: PASSED")
102 test_counter[0] += 1
103
104 # Results Table
105 with doc.create(Section("Tests", numbering=False)):
106     doc.append("Each test conducted is listed below along with its
107         result:")
108     doc.append(NoEscape(r'\newline\nnewline'))
109
110     with doc.create(Tabular('|l|l|')) as table:
111         table.add_hline()
112         table.add_row(('Check Performed', 'Result'))
113         table.add_hline()
114         for entry in diagnostics_log:
115             if ' - ' in entry:
116                 test_name, result = entry.split(":")
117                 table.add_row((test_name.strip(), result.strip()))
118                 table.add_hline()

```

```
109     # Compile
110     doc.generate_pdf('osdag_diagnostics_report', compiler=PDFLATEX_PATH
111                     , clean=True, clean_tex=False)
%----- end code -----
```

## 5.6 Task 3: Report Generation System

In Test 4 of the testing script, it generates a .pdf by the name of test4\_latex\_testing\_suite. This test checks for the basic import and working of the essential .sty files. The generation of the .tex file can be toggled True/False using the clean\_tex flag from doc.generate\_pdf() function.

In the script, in Test 5 final report generation is handled by the function test\_compile\_latex\_with\_pylat. The final report named osdag\_diagnostics.pdf is stored in the same folder from wherever the script is run.

Has a metadata table which lists the Testing Framework, Date and Time, Computer Name and the current User's name logged-in in the computer.

Has the final test results, which mentions all the files it checks for and whether the test passes or fails.

## Osdag Diagnostics Report

June 15, 2025



Test	LaTeX Testing Suite
Date	June 15, 2025
Computer Name	Caesar-MK42
User	Steve

### Checks Done:

The current LaTeX testing suite checks for the presence of the following dependencies:

- Presence of pdflatex.exe either in system path or present in osdag\_latex\_env/bin/windows
- Presence of latexmk.exe
- Presence of the essential .sty files. Also includes all .sty files used by any .tex file generated by Osdag currently for any module.
- Calls each of the essential .sty files and compiles a sample report where each package is used. If compilation is successful it deletes the .tex file and resultant .pdf, .aux, .log. If compilation is unsuccessful, the test fails i.e either a .sty package is corrupt or it is missing.

### Tests

Each test conducted is listed below along with its result:

Check Performed	Result
Test 1 - pdflatex availability	PASSED
Test 2 - latexmk presence	PASSED
Test 3 - lastpage.sty	PASSED
Test 4 - parskip.sty	PASSED
Test 5 - kvoptions.sty	PASSED
Test 6 - ltxcmds.sty	PASSED
Test 7 - kvsetkeys.sty	PASSED
Test 8 - geometry.sty	PASSED
Test 9 - needspace.sty	PASSED
Test 10 - multirow.sty	PASSED
Test 11 - colortbl.sty	PASSED
Test 12 - fancyhdr.sty	PASSED
Test 13 - latexmk.pl	PASSED
Test 14 - amsmath.sty	PASSED
Test 15 - article.sty	PASSED
Test 16 - color.sty	PASSED
Test 17 - fontenc.sty	PASSED
Test 18 - graphicx.sty	PASSED
Test 19 - inputenc.sty	PASSED
Test 20 - lmodern.sty	PASSED
Test 21 - longtable.sty	PASSED
Test 22 - tabularx.sty	PASSED
Test 23 - xcolor.sty	PASSED
Test 24 - LaTeX compilation with all .sty files	PASSED
Test 25 - Osdag Diagnostics Report compilation	PASSED

Figure 5.1: Osdag Diagnostics Report

# Chapter 6

## Conclusions

### 6.1 Tasks Accomplished

#### 1. `osdag-latex-env` – `texmf` Tree Structure

- Set up a custom `texmf` tree for Osdag’s LaTeX environment under `osdag-latex-env/texmf-dist`.
- Organized all required `.sty` and related files into proper CTAN-compliant directory structure (e.g., `tex/latex/`, `fonts/`, `bibtex/`).
- Configured environment variables like `TEXMFHOME` and `TEXINPUTS` to prioritize this custom tree when compiling LaTeX documents.
- Validated the working of packages like `amsmath`, `graphicx`, `fancyhdr`, and others through controlled `.tex` document tests.

#### 2. **Tex Manager – Maintainer’s Manual** Authored a step-by-step Maintainer’s Manual documenting:

- Initial git setup, cloning, and branching practices and to check for `.sty` updates and sync them into the `texmf-dist`.
- To rebuild or refresh the package index using TeX commands. Guidance on versioning, backups, and structure maintenance.
- Added instructions for verifying LaTeX compilation and troubleshooting package errors using test scripts.

**3. Pytest – LaTeX Testing Suite** Built a comprehensive testing suite using pytest to:

- Check the presence of `pdflatex` and `latexmk` in either system path or the Osdag LaTeX env. Confirm availability of essential `.sty` files using recursive directory search.
- Compile a dummy `.tex` document that imports and uses each `.sty` file to ensure runtime compatibility.
- Generate a detailed Diagnostics Report using `pylatex`, complete with metadata, headers, logos, test results, and structured output.
- Implemented custom headers with dynamic date and time, logos using `StandAlone-Graphic`, and fancy table formatting using `colortbl`, `array`, and `longtable`.
- Ensured consistent output whether script was run via IDE or `cmd`, by correcting path handling and resolving image inclusion issues.

## 6.2 Skills Developed

### LaTeX and TeX System Proficiency

- Gained deep understanding of the TeX typesetting system and LaTeX package management.
- Worked with custom `texmf` trees, setting up and organizing `.sty` files in a CTAN-compliant structure.
- Mastered LaTeX compilation commands (`pdflatex`, `latexmk`) and integration with Python via `pylatex`.
- Gained experience in dynamic document generation with logos, styled headers, custom tables, and automated reports.

### Python Development

- Developed Python-based tools for LaTeX diagnostics and reporting using `pylatex`, `os`, `subprocess`, and environment variables.



- Used parameterized testing with `pytest.mark.parametrize` to validate each `.sty` file systematically.
- Managed cross-platform compatibility (IDE vs. CMD) by resolving path resolution, environment setup, and subprocess management.

## **Automation and Testing**

- Built a modular test suite using `pytest` to automate validation of LaTeX environment components.
- Handled test logging and custom assertion messages to improve debugging efficiency.
- Ensured reliable generation and cleanup of `.tex`, `.pdf`, `.log` files during testing.

## **Git and Version Control**

- Practiced clean Git workflows—branching, committing, and syncing changes to the LaTeX environment repo.
- Learned how to track updates to third-party `.sty` packages and maintain version integrity.

## **Documentation and Technical Writing**

- Authored a Maintainer’s Manual with precise, step-by-step instructions, ensuring ease of adoption for future contributors.
- Improved clarity and structure in writing for technical guides and diagnostic logs.

## **Problem Solving and Debugging**

- Diagnosed complex LaTeX compilation errors and path issues under different environments (IDE vs. CMD).
- Used logical strategies to isolate issues (e.g., broken `.sty` packages, image resolution, header formatting).

# Chapter A

## Appendix

### A.1 Work Reports

		Record of Work - Semester Long Internship 2025 - Steve Sojan	
Name :	Steve Sojan		
Project:	Osdag		
Internship:	Semester Long Internship 2025		
Date	Day	Task	Duration of Work Hours
10/02/25	Monday	Orientation   Latest Version of Osdag Installed   Watched Videos of Osdag Installation from User Perspective	2
11/02/25	Tuesday	Finding the bugs I encountered while preparing the screening task, listing them out   Meeting at 18:45	3
12/02/25	Wednesday	Meeting at 19:15   Brainstormed new solutions to add to the installer and resolve current issues (stuck installer)	3
13/02/25	Thursday	Possible Solutions to a stuck installer - Izma compression method or unpacking in chunks   Worked on Assigned Module	3
14/02/25	Friday	Worked on Task - 0   Assigned Cleat Angle Connection Module   Understanding the codebase	5
15/02/25	Saturday	Continued Documentation on Cleat Angle Connection Module   imports, UI Functions, Calculation Functions, CAD functions	8
16/02/25	Sunday	Designed the Final Flowchart for documenting the working of Cleat Angle Connection Module	3
17/02/25	Monday	Discussion on Task - 0 at 18:15	3
18/02/25	Tuesday	Documentation Team Meeting for Tension Member Bolted Connection   Roadmap   imports discussion scheduled for Thursday	3
19/02/25	Wednesday	Git/Github Usage Policy Meeting   Went through the imports section of Tension Member Bolted Connection in detail	3
20/02/25	Thursday	imports discussion for Tension Member Bolted connection - Meeting at 19:30	3
21/02/25	Friday	Installer Team Meeting   Discussed the Tex packages Osdag needs   Reported the bugs I found and packages inherent with TinyTex	3
22/02/25	Saturday	Finding the cause/ solution to the Latex Creation Error bug	3
23/02/25	Sunday	WEEKLY OFF	0
24/02/25	Monday	Tried to look for the cause of the bug in <a href="#">ui_summary_popup.py</a> and <a href="#">report_generator_latex.py</a> files	2
25/02/25	Tuesday	Found the cause for the bug within the source .tex file - a backslash prior to each '.' to escape the subscript operator	4
26/02/25	Wednesday	Started work on creating a localized tex engine, that can call userpackages w/out the need of a Tex distro	4
27/02/25	Thursday	Listing out all packages, dependencies, macros and binaries	2
28/02/25	Friday	Designing the texmf-tree, specifying LaTeX to look for the packages in the directory osdag-latex-env	3
01/03/25	Saturday	Making respective changes in <a href="#">SectionModeller_Latex.py</a> and <a href="#">reportGenerator_latex.py</a>	4
02/03/25	Sunday	Making respective changes in SectionModeller_Latex.py and <a href="#">reportGenerator_latex.py</a> for Osdag to use the texmf-tree to generate pdf reports	3
03/03/25	Monday	EXAM BREAK	0
04/03/25	Tuesday	EXAM BREAK	0
05/03/25	Wednesday	EXAM BREAK	0
06/03/25	Thursday	EXAM BREAK	0
07/03/25	Friday	EXAM BREAK	0
08/03/25	Saturday	EXAM BREAK	0
09/03/25	Sunday	EXAM BREAK	0
10/03/25	Monday	EXAM BREAK	0
11/03/25	Tuesday	EXAM BREAK	0
12/03/25	Wednesday	Revising changes in SectionModeller_Latex.py and <a href="#">reportGenerator_latex.py</a>	4
13/03/25	Thursday	Meeting at 19:00- 19:30 today to decide on packaging Tex locally or with a Tex distro   Read up on available Tex Distro	3
14/03/25	Friday	Documenting the process of creating a localized Tex engine	3
15/03/25	Saturday	Completing documentation and submission	2
16/03/25	Sunday	WEEKLY OFF	0
17/03/25	Monday	Discussed & Reviewed the installer made for the screening task	4
18/03/25	Tuesday	Meeting at 17:00   Tested the installer created for the screening task along with the team	4
19/03/25	Wednesday	Discussions on making an offline installer	4
20/03/25	Thursday	Completed Documentation on the Localized Tex Engine	4
21/03/25	Friday	Research & find ways to keep the TeX packages updated   Transition from being localized to updated	4
22/03/25	Saturday	Meeting at 17:45   Assigned to integrate GitHub Submodules	4
23/03/25	Sunday	WEEKLY OFF	0
24/03/25	Monday	Meeting at 20:00   Discussed updates on the revised installer	5
25/03/25	Tuesday	Reserached ways to keep Tex packages updated	4
26/03/25	Wednesday	Started documentation on keeping Tex packages updated	4
27/03/25	Thursday	Researched Git Submodules as an alternative to keeping the Tex pacakages updated	4
28/03/25	Friday	Completed documentation on approaches to keep the tex packages updated	4
29/03/25	Saturday	Designed the pipeline of keeping the tex packages updated using Github Submodules	4
30/03/25	Sunday	WEEKLY OFF	3
31/03/25	Monday	Pruning the texmf tree   Removed tipkg	0
01/04/25	Tuesday	Started creating a custom texmf tree with the 11 tex packages as mentioned on the channel	4
02/04/25	Wednesday	Debugging each error and adding missing packages manually	4
03/04/25	Thursday	Meeting at 18:00   Discussed on building Osdag's own texmf tree environment	4
04/04/25	Friday	Filtering the .sty packages that we need	4
05/04/25	Saturday	Monitoring the final size of the Tex Enironment from 300 MB originally to 173 MB	4
06/04/25	Sunday	WEEKLY OFF	0
07/04/25	Monday	Started work on creating a Maintainer's Manual	4
08/04/25	Tuesday	Designed a pipeline for keeping Tex packages updated using Git submodules using Tex Manager and Git Submodules by a Maintainer	4
09/04/25	Wednesday	Pruned the final texmf tree and pushed to Git repo - osdag-latex-env	4
10/04/25	Thursday	osdag-latex-env is referred to as an external repo using Github Submodules to maintainer's local repo	4
11/04/25	Friday	Current script removes the .tex file, found the solution that retains the .tex file while compiling using pylatex	4
12/04/25	Saturday	Briefed up on how Git Submodule references a certain commit	3
13/04/25	Sunday	Initialize git in TinyTex directory and create .gitignore file.	4
14/04/25	Monday	Configure .gitignore to ignore all files except: bin, texmf-config, texmf-dist, texmf-local, and texmf-var.	4
15/04/25	Tuesday	Added the external repository (tex-env-submodule) as a remote to the local TinyTex git repository.	5
16/04/25	Wednesday		4
17/04/25	Thursday	Push the filtered texmf tree to the external repo preserving TinyTex's original structure.	4
18/04/25	Friday	Tested the new fully online installer that the team had devised	4
19/04/25	Saturday	Cloned osdag-admin/Osdag repo and add the submodule (tex-env-submodule) to the intended path (src/osdag/data/ResourceFiles/osdag-latex-env)	4
20/04/25	Sunday	WEEKLY OFF	3
21/04/25	Monday	Validate and edit the .gitmodules file in the main repo to confirm the submodule path and URL are correct.	4
22/04/25	Tuesday	Use tlmgr update --list to check for available package updates inside the maintainer's TinyTex.	4
23/04/25	Wednesday	Ran a full test run of pipeline using Git Submodules and Tex Manager with existing packages	4
24/04/25	Thursday	Updated packages and pushed to repo referred to as submodule	4
25/04/25	Friday	Ran a complete test run with updated packages and updated commit reference successfully	5
26/04/25	Saturday	EXAM BREAK	0
27/04/25	Sunday	EXAM BREAK	0
28/04/25	Monday	EXAM BREAK	0
29/04/25	Tuesday	EXAM BREAK	0
30/04/25	Wednesday	EXAM BREAK	0

01/05/25	Thursday	EXAM BREAK	0
02/05/25	Friday	EXAM BREAK	0
03/05/25	Saturday	EXAM BREAK	0
04/05/25	Sunday	EXAM BREAK	0
05/05/25	Monday	EXAM BREAK	0
06/05/25	Tuesday	EXAM BREAK	0
07/05/25	Wednesday	EXAM BREAK	0
08/05/25	Thursday	EXAM BREAK	0
09/05/25	Friday	EXAM BREAK	0
10/05/25	Saturday	EXAM BREAK	0
11/05/25	Sunday	EXAM BREAK	0
12/05/25	Monday	Completed documentation for the Tex Manager - Maintainer's Manual Approach and shared the same.	4
13/05/25	Tuesday	Set up the initial LaTeX test environment and verified pylatex installation with basic compilation.	4
14/05/25	Wednesday	Designed a minimal .tex document and tested manual compilation using pdflatex from command line.	4
15/05/25	Thursday	Explored the structure of osdag-latex-env and located the internal pdflatex.exe	4
16/05/25	Friday	Developed functions to detect pdflatex in system PATH or fallback to osdag-latex-env	5
17/05/25	Saturday	Integrated conditional logic to assign the correct path to PDFLATEX_PATH	4
18/05/25	Sunday	WEEKLY OFF	0
19/05/25	Monday	Created pytest tests to verify existence of pdflatex and fallback behavior	5
20/05/25	Tuesday	Listed and confirmed core .sty files required by the LaTeX compiler	4
21/05/25	Wednesday	Implemented pytest.mark.parametrize to automate checks for all required .sty files.	4
22/05/25	Thursday	Added a specific test for verifying presence of latexmk.exe in the internal LaTeX environment	4
23/05/25	Friday	Manually constructed a diagnostic .tex file using all required packages.	5
24/05/25	Saturday	Programmed the compilation of this .tex file via pylatex.Document with a custom compiler path.	4
25/05/25	Sunday	WEEKLY OFF	0
26/05/25	Monday	Handled CompilerError exceptions during LaTeX execution and added verbose error reporting.	4
27/05/25	Tuesday	Refined file cleanup operations post-compilation for .pdf, .log, .aux, and .tex.	4
28/05/25	Wednesday	Applied Arrange-Act-Assert (AAA) testing methodology to all existing test cases.	6
29/05/25	Thursday	Validated that all tests passed under both system and bundled pdflatex configurations.	4
30/05/25	Friday	Optimized the codebase for modularity, reusability, and logical flow of execution.	5
31/05/25	Saturday	Finalized a diagnostic framework for LaTeX compilation in Osdag	4
01/06/25	Sunday	WEEKLY OFF	0
02/06/25	Monday	Incorporated testing logic, if all tests should pass - script returns 0 else even if 1 test fails return 1.	4
03/06/25	Tuesday	Designed the struture of a Diagnostics Framework Report that lists the status of all tests using pylatex	4
04/06/25	Wednesday	Worked on report generation system of the LaTeX testing suite	5
05/06/25	Thursday	Meeting at 19:00   Shared updates on LaTeX testing suite   Further tune testing and Diagnostics Report Compilation	4
06/06/25	Friday	Tuned the Osdag Diagnostics Report to use packages from osdag-latex-env for generating reports	5
07/06/25	Saturday	Added custom Osdag header and footer to the Diagnostics Report Generation	4
08/06/25	Sunday	WEEKLY OFF	0
09/06/25	Monday	Worked on the Final Developer Manual that combines work from the entire Installer Team	4
10/06/25	Tuesday	Experimented with calling osdag-latex-env as a module to make minimal changes in codebase	4
11/06/25	Wednesday	Started drafting the report for designing the LaTeX Testing Suite through a user-friendly guide	5
12/06/25	Thursday	Completed documentation that describes the entire LaTeX testing suite   Bugs fixed suggested during mentor review	4
13/06/25	Friday	Finalized layout of the Diagnostics Report header and metadata table formatting	5
14/06/25	Saturday	Integrated remaining unused packages into the report with PyLaTeX usage examples	5
15/06/25	Sunday	WEEKLY OFF	0
16/06/25	Monday	Final review and started work on the Final Internship Report	4
17/06/25	Tuesday	Prepared Final Internship Report and compiled all work done during the report	4
18/06/25	Wednesday	Final meeting with mentor to discuss internship performance and next steps	5
19/06/25	Thursday	Presented the final internship report to the mentor	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.