Summer Fellowship Report

On

Osdag on Cloud

Submitted by

Annepu Sai Charan

Under the guidance of

Prof.Siddhartha Ghosh
Civil Engineering Department
IIT Bombay

Under the mentorship of

Nagesh Karmali
Danish Ansari

August 23, 2023

Acknowledgment

I would like to extend my heartfelt gratitude towards FOSSEE for providing me with an invaluable platform to delve into a realm that greatly captivates me – the art of building tools. The opportunity granted to me by FOSSEE is truly a privilege, and I wish to convey my sincere thanks to every individual who played a part in conceiving and executing the meticulous selection process, which was centered around insightful screening tasks. It is due to their efforts that I now find myself an integral part of a dynamic team that wholeheartedly advocates for open source software and its profound significance.

I thank all the Osdag members, who are wonderful mentors and great team. I extend my warm thanks to Nagesh Karmali, whose role as a Project Research Associate has been instrumental, as well as to Danish Ansari, the Assistant Project Manager, and the entire team. Who made us feel welcome and planned all the tasks meticulously during this period. I am grateful that I got a chance to work under Prof. Siddhartha Ghosh who took time to mentor us and monitored individual contributions as well.

In conclusion, my appreciation knows no bounds for the opportunities, guidance, and support I have received from FOSSEE, the Osdag community. This journey has not only allowed me to explore my passions but has also equipped me with invaluable skills and experiences that will undoubtedly shape my future endeavors.

Contents

Chapter 1

Introduction

This chapter provides a comprehensive overview of both "Osdag" and it's cloud-based counterpart "Osdag on Cloud," while also highlighting the distinctions between these two products. Additionally, it offers a succinct overview of the intended audience for these solutions.

1.1 What is Osdag?

"Osdag" is a trans-formative exemplar of Free/Libre and Open Source Software (FLOSS), meticulously crafted to redefine steel structure design. Rooted in Python, this software embodies collaborative excellence and technical finesse. The integration of PythonOCC, a 3D CAD modeling framework, elevates "Osdag" beyond the ordinary. By intertwining code with visual representation, the software offers an immersive platform for users to engage with and refine their structural concepts.

The "Share alike" policy, advocated by FOSSEE, adds a layer of significance. Beyond open source principles, it envisions a perpetually evolving software landscape, aligned with industry and academic needs. In essence, "Osdag" transcends software conventions, embodying a collaborative journey that empowers engineers, architects, and learners. In a rapidly changing world, it stands as a beacon of progress, fueled by the ethos of open source collaboration.

1.2 Challenges with the Osdag software.

While undoubtedly a valuable tool, the "Osdag" software is not without its challenges. One prominent obstacle is the time-intensive nature of setting up the desktop application on local systems, a concern shared by both developers and users. The process can often be compounded by potential bugs stemming from version disparities or operating system variations. This complexity can lead to a range of issues during setup. Enterprises into this backdrop, "Osdag on Cloud" emerges as a promising solution.

1.3 What is Osdag on Cloud?

"Osdag on Cloud" is an innovative approach circumvents the intricate setup process altogether, offering a streamlined alternative. By shifting the software's operations to the cloud, users and developers can sidestep compatibility issues and operating system nuances. This forward-looking initiative not only enhances convenience but also empowers seamless accessibility, ushering in a new era of user experience.

"Osdag on Cloud" represents the cloud-based iteration of Osdag built with *React.js* and *Django*. With this version, users are relieved of the necessity to download the Osdag desktop application onto their computers. Instead, they can readily harness the complete spectrum of Osdag features by simply navigating to the website. This streamlined approach allows users to access the software's functionalities without any local installations. Notably, intricate tasks such as computations and report generation, design computations, etc. are seamlessly handled within the cloud environment, further enhancing user convenience.

1.4 Who can use Osdag on Cloud?

"Osdag on Cloud" is created for both educational purposes and industry professionals. As FOSS is currently funded by MHRD, the Osdag team is developing the software in such a way that it can be used by students during their academics to provide them with a better insight into the subject. Osdag is designed to be usable by anyone, from novices to professionals. Its simple user interface makes it more flexible and attractive than other software options. Video tutorials are available to help users get started. You can access the video tutorials for Osdag here.

Chapter 2

About My Work

This chapter gives the gist the work I have done during my fellowship. The next chapter will describe the work briefly.

- Orchestrated the design and development of the foundational API for the Osdag server, ensuring robust functionality and optimal performance.
- Employed the "reducer + context" pattern in React to architect and establish a sophisticated global state management system, enhancing the efficiency and maintainability of the application.
- The API integration in several parts of the project.
- Developed the output calculation functionality which uses "Osdag Desktop" application code.
- Contributed to the refinement of the design report functionality and refining the user experience.
- Developed the save output functionality which stores the output with essential metadata in a structured CSV format.
- Designed and developed design preference functionality end-to-end.
- Developed the "saving custom material" functionality in the browser storage.
- Implemented the download log message (osi format) and load input functionality.

Chapter 3

Osdag on Cloud

This chapter delineates the characteristics and implementation of the "Osdag on Cloud." It provides a concise overview of the aspects covered in the preceding chapter, encompassing a discussion of various APIs developed, as well as the underlying logic within the client-side component of the application

3.1 The root API of osdag server.

The list of connection types and there sub-types are obtained by the root API. The root API and the client is built and integrated in such a way that even if we add more sub-types (up-to 3 levels including leaf), it automatically detects and UI is rendered accordingly and it functions as desired. More details on the root API can be found here. The hierarchy is maintained while building the API.

3.2 The state management of the root data.

The Context API and Reducer from React.js are utilized for the better state management of the root API data. Since the data is used in multiple components maintaining a global state is efficient and prop drilling can be avoided. There are exactly two files which contains the code for the same:

- **GlobalState.jsx**: This file contains the state and the actions.
- **AppReducer.jsx**: This file contains the reducers for updating the state from GlobalState.jsx.

The actions and reducers are self explanatory.

3.3 Input fields population.

The Context API and Reducer is again utilized. It has separate file for input data handling. The code can be found in ModuleState.jsx and ModuleReducer.jsx. The following fields comes form the database:

- Supported and supporting section.
- Material.
- Diameter.
- Property Class.
- Plate thickness.

The connectivity also comes from the server but it is not listed in the database.

3.4 Output Calculation.

This section will describe the client and server side of output calculation. The action for API call is in the ModuleState.jsx file and reducer is in ModuleReducer.jsx. The back-end code is in the outputCalc_view.py which is inside osdag/web_api.

3.4.1 Client side of output calculation.

The inputs fields are checked so that no missing value is passed to the server, the the output API is called and cookie is also attached in the API call. The response contains the output in such a manner that the output can be divided into four subsections, which are the following:

- Plate.
- Bolt.
- Section Details.
- Weld.

The capacity and spacing have pop-ups in order to save some of the space and to show the most important fields. The zero value fields are not shown in the UI. Some of the input values are obtained from the design preference which is described here.

3.4.2 Server side of output calculation.

The input values, module id and cookie is store in Design table of database, this is later used in report generation. Under the hood the view of output calculation uses core finplate module which is used by the "Osdag" desktop application. The logs are also obtained by the finplate module and combined and formatted and then sent to client.

3.5 Design report functionality.

This section will describe the working of design report functionality.

3.5.1 Client side of design report functionality.

There are mainly two APIs for the generation and downloading of the design report. The first API call generates the report and the second call one gets the PDF of the report. The actions are located in ModuleState.jsx. The following actions calls the API:

- **createDesignReport**: This actions sends the metadata to the server along with the cookie ID. The API returns the report id.
- **getPDF**: This function uses the report id to get the design report from the server.

3.5.2 Server side of design report functionality.

The cookie id is used to retrieve the output and other required values from the Design table of the database. Again the core finplate module of "Osdag" desktop application is used to generate the latex file of the design report. The report is saved in the file_storage/design_report directory. The *pylatex* module from python is utilized. This view returns the report id to the client.

The second API call from the client sends the report ID and this is used by the get method of the view. It locates the latex file and uses *pdflatex* module to convert it into the PDF file and it sends to the client.

3.6 Save output functionality.

Once the design is being successfully created the user has the option of saving the output. The output is saved in the CSV format. It contains the other information such as module name. The functionality is resides the "saveOutput" function which is location in FinPlate.jsx file. The core code is being shown below:

```
Object.keys(output).map((key, index) => {
  Object.values(output[key]).map((elm, index1) => {
    data[key + '.' +
         elm.label.split(' ').join('_')] = elm.val
  })
})

data = convertToCSV(data)
const csvContent =
    'data:text/csv;charset=utf-8,'
    + encodeURIComponent(data);
const link = document.createElement('a');
link.setAttribute('href', csvContent);
link.setAttribute('download', 'output.csv');
document.body.appendChild(link);
```

```
link.click();
document.body.removeChild(link);
```

3.7 Design Preference.

The design preference is one of the core functionality of the Osdag. The user can change some of the default values of the inputs which depends upon the module. The following sub-sections will brief about the design preference.

3.7.1 Client side of the design preference.

The design preference can be accessed from the *edit* tab from the menu. The design preference modal has the following subtabs and functionality are as follows:

- **Column section/primary beam**: This section displays the *supporting* section data such as dimensions and section property. All the data comes from the data. The supporting section material has the default value which is "E 165 (Fe 290)" and it can be changed. The other data such as yield strength and ultimate strength are coming from the database which depends upon the material. The *type* is "Welded" for custom material and "Rolled" for other materials. Similarly *source* is "IS808_Rev" for normal material and "Custom" for custom materials.

- **Beam section/secondary beam**: This section displays the *supported* section data such as dimensions and section property. All the data comes from the data. The supported section material has the default value which is "E 165 (Fe 290)" and it can be changed. The other data such as yield strength and ultimate strength are coming from the database which depends upon the material. The *type* is "Welded" for custom material and "Rolled" for other materials. Similarly *source* is "IS808_Rev" for normal material and "Custom" for custom materials.

- **Connector**: This section displays the mechanical details of the connector material such as ultimate strength and yield strengths. The connector material has the default value of "E 250 (Fe 410W)A".

- **Bolt**: This section allows user to edit bolt type, bolt hole type and slip factor. The default values are "Pre-tensioned", "Standard" and "0.3" respectively.

- **Weld**: This section allows user to edit "Type of weld fabrication" and "Metal grade" which is set to "Shop Weld" and "410" respectively.

- **Detailing**: This section allows user to edit "edge preparation method", "gap between beams" and "are member exposed to corrosive influence", the defaults is begin set to "Rolled machined flame cut", "10" and "No" respectively.

- **Design**: This has only one field “Design method” which can not be changed as for time being”

The data is stored in ModuleState.jsx and defaults values are also stored their. The user can click “Save and Continue” from the bottom to save the values, if user tries to closes the modal or clicks default then they will receive a warning. They can either proceed or save the changes. For the implementation there is difference state which is being rendered in design preference modal after saving the values are updated in the inputs”

3.7.2 Server side of design preference.

The server returns the details of the supported and supporting section details and material mechanical properties of connector, supported and supporting section materials.

The API for saving custom material is created but not fully implemented. The work is going on for this one.

3.7.3 Custom material.

The custom material can be created by clicking on *Custom* under the drop-down of materials. It opens up a pop-up and ultimate strength and yields strength can be inputted. Some of the important points to keep in mind:

- The yield strength and ultimate strength should be in the range of *165 to 1500*.
- The naming of the custom material is special. It should be in the form of *Cus_<Fy_20>_<Fy_20_40>_<Fy _40>_Fu*. The Fy and fu are as follows:
  - **Fy_20**: Yield strength less than 20 Mpa.
  - **Fy_20_40**: Yield strength greater than 20 Mpa and less than 40 Mpa.
  - **Fy_40**: Yield strength greater than 40 Mpa.
  - **Fu**: Ultimate strength.

All the fields are required. The unauthenticated user can save only in browser storage but the authenticated user has the option of storing data in both browser storage and in the database. The Figure 2.1 shows the custom section.

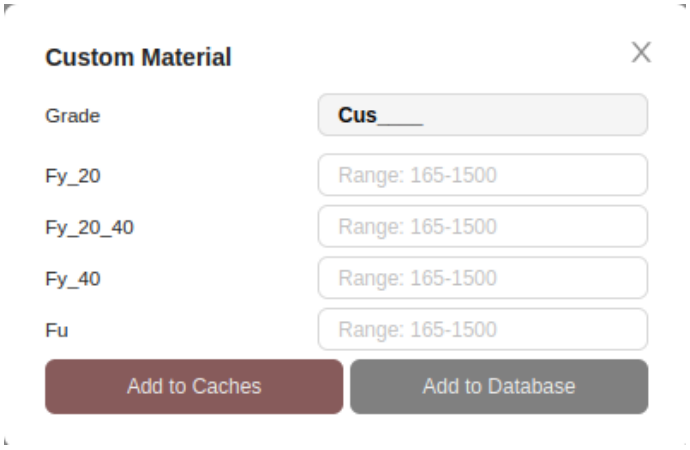

Figure 3.1: Custom Section

3.8 Download input and log messages.

This options can be found in *file* tab of the menu. The input can be downloaded in *.osi* format and can be loaded into the application. The file also contains the fields from the design preferences. The input is getting downloaded in the specific format and it shouldn't be changed so that it can be loaded properly.

The log messages can also be downloaded. It can be used as a reference for the specific input.

3.9 Load input functionality.

The osi file which is saved can be loaded to automatically fill the input values. If the downloaded file is tampered then there will be problem in loading the input file. For now it does not handles all the scenarios.

Reference

- Osdag Repository
- Osdag Website
- Osdag Sample Inputs
- React v18
- Django Rest Framework (DRF)
- Antd and React
- React: Scaling with Reducer and Context API
- PostgresSQL Downloads
- PostgresSQL Documentation