Journal of Energy

VOLUME 75 Number 3 | 2026

journal homepage: http://journalofenergy.com/

Advanced Scalable WebApp Approach for Automated Simulation of Transformer Components Using Python

Harikrushna Dodiya, Irene Woyna, Luigi De Mercato

Summary **—Accelerating transformer development lifecycles through simulation automation requires the adoption of new technologies that reduce manual effort and democratize simulation workflows. Traditional approaches to simulation automation often involve multiple local software installations, require specialized skills, lack centralized version control, and do not offer unified database storage. To address these limitations, this paper presents a novel technology: a web-based application developed in a Python environment to enable streamlined access to simulation automation. Python's extensive ecosystem including AI, ML, NumPy, Pandas, and all PyAnsys modules facilitate seamless integration of current and future technologies within this interface. With this Pythonic web application, users can easily input data and visualize 3D simulation results across devices such as mobile phones, laptops, and workstations. The newly developed interface supports automation of complex non-linear Multiphysics simulation workflows, parameter input, in-browser 3D result visualization, and automatic report generation. This technology is applicable to any type of transformer or its components.**

Keywords **— Advance simulation method, WebApp, Python, Multiphysics.**

I. Introduction

In today's situation, the demand for transformers is rising due to increasing energy needs, rapid product development, and sustainability requirements. The transformer design is quite time consuming, as the customer asks for simulation validation of seismic, vibration, transportation, short circuit, and other loading conditions. The traditional development cycle uses a physical prototyping and manual simulation workflow for every design, which leads to an increase in the product development cycle. As today's energy market is booming, there will be increased pressure on transformer manufacturers to reduce product life cycle development without compromising quality and reliability. This will lead to the use of advanced digital tools, which will replace the traditional development cycle.

The simulation-driven product development lifecycle is the best modern engineering practice due to a less expensive and faster reliable product launch and allows for the virtual testing of complex Multiphysics phenomena, and their interactions such as electromagnetic, structural and thermal inside transformers. However, traditional manual simulation is a cumbersome process. Even traditional automation requires installing multiple software and its dependencies (Ansys Customization Toolkit (ACT), Ansys Parametric Design Language (APDL) script) on every computer when a user wants to run the automation. So, it will not allow good collaboration between teams. The version control is quite challenging with different locations. Also, integration with modern architecture is difficult. The lack of having a centralized version control system and common cloud data storage, there will be inconsistencies and errors while handling versions and data. This will directly impact the product development cycle.

To overcome these issues, there is a new way to handle traditional automation without changing the fundamentals of FEA. It is possible through automation using the Pythonic interface, utilizing the PyAnsys modules of Ansys. From this, we can develop a web-based system and efficiently perform simulations automation using High-Performance Computing (HPC) and other latest technologies. The current web-based app technology software is not flexible, and it is not open source, so it will be expensive. Here, we are using open-source technology for web app development. Due to that, the cost of the platform is free to use.

This paper showcases a new simulation automation methodology for development of a web application framework using Pythonic applications utilizing the Python based WebApp modules and PyAnsys [1] modules, aiming at simplifying and shortening the simulation development cycle of transformers. The new methodology combines all the capabilities available in Python World [2], from artificial intelligence and machine learning libraries to scientific computing tools and various Finite Element Analysis (FEA) software. This web-based application is user-friendly because it requires no complicated setup on local computers, allows 2D and 3D visualization from any browser or mobile system and supports automatic reporting for customer submission.

II. Guidelines For Manuscript Preparation

This section provides an outline of the architecture and its integration into the Python ecosystem. The development of transformers needs a strong, cybersecurity and versatile simulation platform with a Web-based user interface. It can deal with complex Multiphysics simulation automation while remaining user-friendly to users of different skill levels and different global locations. The proposed methodology addresses these needs by introducing a Python-based web application that integrates advanced simulation capabilities using PyANSYS into an innovative user interface.

(Corresponding author: Harikrushna Dodiya)

Harikrushna Dodiya, Irene Woyna and Luigi De Mercato are with GLOMUS Network, Hitachi Energy (email: harikrushna.dodiya@hitachienergy.com, irene.woyna@hitachienergy.com, luigi.de-mercato@hitachienergy.com)

Harikrushna Dodiya, Irene Woyna, Luigi De Mercato, Advanced Scalable WebApp Approach for Automated Simulation of Transformer Components Using Python, Journal of Energy, vol. 75 Number 3 (2026), 85–87
https://doi.org/10.37798/2026753791

© 2026 Copyright for this paper by authors. Use permitted under Creative Commons Attribution-NonCommercial (CC BY-NC) 4.0 International License

A. Architecture Overview

The proposed methodology is developed as a web application using Python with the WebApp module. This platform allows for simulation workflow management using PyAnsys of different modules and avoiding multiple installations. The framework has two main parts as shown in figure 1:

- Frontend: A user friendly and intuitive interface made with Python based modules module in python to enter parameters and run the simulation.
- Backend: Python scripts which handles tasks such as execution of simulations and integration with other tools like database management, Login and other HPC systems

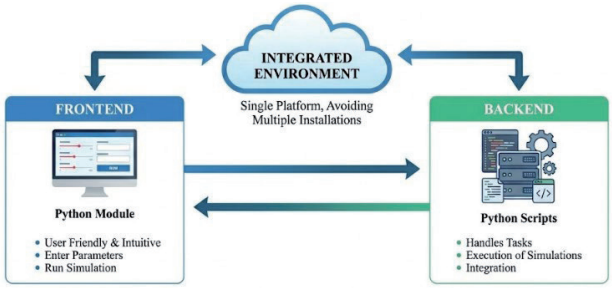


Fig. 1. Architecture overview showing the interaction between Frontend and Backend.

B. Integration with Python Ecosystem

The Python programming language has a vast set of modules that increase the effectiveness of the proposed platform:

- Scientific Computing: Use of NumPy [3] and Pandas [4] for data manipulation.
- Visualization: Plotly [5] and Matplotlib [6] for interactive and static visualizations of 2D and 3D results files.
- Simulation Interfaces: PyAnsys different modules for efficient interaction with the solvers in Ansys.
- AI/ML: TensorFlow and Scikit-learn for machine learning & optimization.

C. Key Functionalities

- Parameter Management: Simplified input through an intuitive web interface. (Excel upload, Parameter input, Radio tab, drop down etc.)
- Workflow Automation: Automatic execution of complex simulation tasks involving Multiphysics.
- In-Browser 3D Visualization: The rendering and visualization of results using 3D visualization python modules.
- Automatic Report Generation: The simulation report summaries can be exported in PDF.
- Global Accessibility: The availability of the program on all platforms ensures accessibility through laptops and workstations across globe

D. Security Considerations

- User Authentication: This WebApp is hosted on Hitachi Energy internal network, so only Hitachi Energy employees are able to access this WebApp and different kinds of automation.

III. Implementation Details

The deployment of the web application platform has solved the issue of an easy-to-use, user-friendly interface, speeding up using HPC, performance improvement, security and scalability. The deployment utilizes Python based webapp module to help engineers develop an interactive dashboard easily. The backend is connected to PyAnsys and other scientific python modules like Numpy, which is responsible for FEA simulations solutions. Plotly and 3D visualization python modules are used to create a 3D contour display in a web browser.

The application follows a modular design:

- Simulation Module: Integration with PyAnsys. currently supporting both static and dynamic analysis simulations as shown in figure 2 and 3 different types of simulations automation.
- User interface Module: This module has the function to provide the user input required in the simulation as show in figure 3.
- Visualization Module: It creates 3D plots and interactive dashboards for result analysis using PyANSYS as shown in figure 4.
- Reporting Module: Enables the automated generation of detailed reports along with charts, tables, simulation snapshots and 3D contour plots.
- Database Module: Data storage and management in a simulation regarding versioning and user profile management.

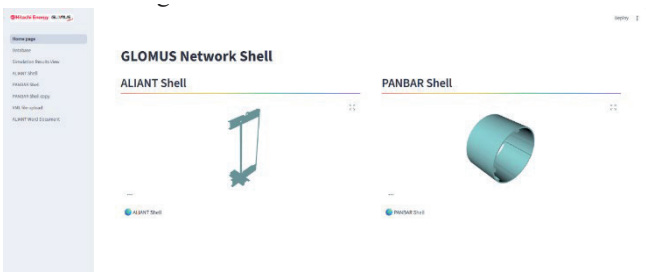


Fig. 2. Home page of the Python-based WebApp displaying different automations.

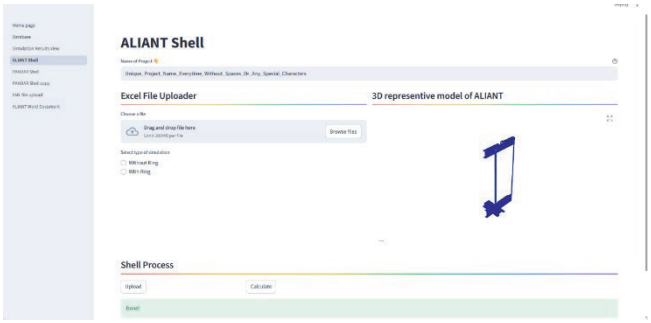


Fig. 3. Automation input module where users upload parameters, configure input selections, and initiate simulations.

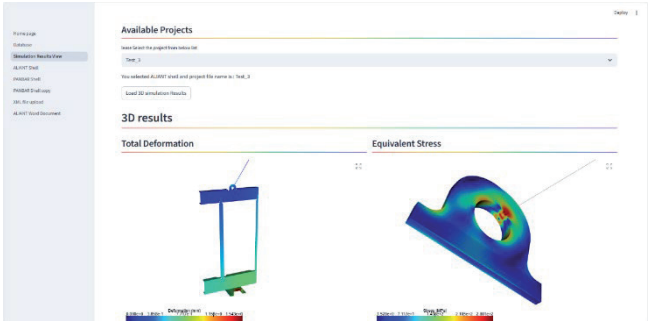


Fig. 4. In-browser 3D visualization showing simulation result contours generated ssthrough Python visualization modules.

Harikrushna Dodiya, Irene Woyna, Luigi De Mercato, Advanced Scalable WebApp Approach for Automated Simulation of Transformer Components Using Python, Journal of Energy, vol. 75 Number 3 (2026), 85–87
https://doi.org/10.37798/2026753791
© 2026 Copyright for this paper by authors. Use permitted under Creative Commons Attribution-NonCommercial (CC BY-NC) 4.0 International License

IV. Results And Discussion

Assessment of the proposed web application methodology built with Python based modules examines how the application can simplify the simulation process and user experience. The Web-based platform increases the user experience and efficiency. The current Web application is tested rigorously in different global locations and in different simulation automations. The current application does not incorporate capabilities to handle artificial intelligence and machine learning yet. The following parameter is the outcome of testing.

A. Performance Improvements

Tests using the Webapp show massive gains in performance:

- Setup Time Reduction: Up to 80-90% compared to traditional workflows since users did not even have to install any software or other dependencies.

Table I

Setup Time Reduction

Workflow Stage	Traditional Duration	Proposed WebApp Duration
Geometry Simplifications	~4 hours	2–5 minutes
Pre-processing (meshing, connections, contacts, etc.)	~3 hours	5–10 minutes
Solution	~3-5 hours	same time
Load-case setup	~3 hours	5–10 minutes
Report generation	~4 hours	2–3 minutes

- Error Reduction: Automated parameter validation reduces the element of human errors.
- Accessibility: Engineers can run simulations on any device without the need for any specialized installation.

B. Scalability and Flexibility

The Web application platform is scalable with some limitations and supports various types of transformers as well as various components with different types of simulation needs. It can be implemented on cloud platform to increase its scalability. However, Python based webapp development module, which is not scalable or fast compared to professional webapp development tools like React, Angular and Vue.

C. Comparative Analysis

In comparison to the conventional process, the solution provides the following advantages:

- Reliable products, based on underlying physics. preliminary assessment of structural strength at bidding stage. Cost optimized products.
- Accelerating product development lifecycles. Reducing development costs. Improving product quality. Increased operational efficiency
- Reduced operational expenses because of open-source software and Global Accessibility
- Improved collaboration via common data storage and Centralized Version Control and Management
- Efficient Utilization of HPC Resources and Eliminate Local Installations.

V. CONCLUSION AND FUTURE WORK

The proposed web application platform in Python is a big leap in the transformer simulation workflow, automating it for various simulations and making it accessible to every company employee across the world. Being open source, it is relatively cost-effective alternative to platform solutions provided by other companies. Its modular architecture is designed to be expandable with new emerging technologies like AI and ML. It is tightly integrated with Python's large ecosystem. The core automation (Static and dynamic simulation) and visualization are done in the current version. It will be further extended in subsequent versions with AI and ML functionalities to unlock predictive modeling and automated optimization. However, the use of Python based WebApp modules has some limitations. This python module is generally less scalable and slower compared to JavaScript based frameworks such as React, Angular or Vue. Based on current study we found that Python based modules can be used by small or medium scale organizations without any issue. In contrast, for large enterprise level companies like Hitachi, Python based modules platforms may not meet the scalability and performance requirement achieved by JavaScript based web application platform.

In the future, other work will involve with help of JavaScript based Webapp tool:

- Multiphysics Simulation: Backend code will be developed for electro-thermal, electro-structure and thermal-structural simulation.
- AI-Driven Optimization: Uses machine learning algorithms to suggest the best design parameters.
- Predictive Analytics: The use of AI models to predict performance and find failure points.
- Cloud Integration: This means using cloud computing and cloud storage to enable high-performance simulations and global scalability

References

[1] Ansys, Inc., »PyAnsys,« Version 0.11, 2026. [Online]. Available: https://docs.pyansys.com/.

[2] Python Software Foundation, "Python Language Reference," https://www.python.org/, 2026.

[3] C. R. Harris et al., "Array programming with NumPy," Nature, vol. 585, pp. 357–362, 2020.

[4] W. McKinney, "pandas: a foundational Python library for data analysis and statistics," Python for Data Analysis, O'Reilly Media, 2017.

[5] Plotly Technologies Inc., "Plotly Open Source Graphing Library," https://plotly.com/python/, 2026.

[6] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, 200.

[7] E. P. Wigner, "Theory of traveling-wave optical laser," *Phys. Rev*., vol. 134, pp. A635–A646, Dec. 1965.ss

© 2026 Copyright for this paper by authors. Use permitted under Creative Commons Attribution-NonCommercial (CC BY-NC) 4.0 International License