Northeastern University

(Computer Science)

Team Indigenous Tech Circle

Green Software Engineering

Author: **Liyao Zhang**
ID: 002837359
E-mail:
zhang.liya@northeastern.edu

Author: **Wei Zhao**
ID: 001969892
E-mail:
zhao.wei2@northeastern.edu

Author: **Jun Liu**
ID: 001611944
E-mail: *liu.jun5@northeastern.edu*

Author: **Ruijin Zhang**
ID: 002622440
E-mail:
zhang.ruijin@northeastern.edu

Tutor: **Yvonne Coady**
Department: Computer Science
E-mail: *m.coady@northeastern.edu*

October 10, 2024

Contents

1 Problem Statement

The rapid expansion of the Information and Communication Technology (ICT) sector is significantly increasing global greenhouse gas emissions, accounting for approximately 2-3% of total emissions. As a core component of ICT systems, software indirectly drives substantial energy consumption through its use of hardware resources. However, there is currently no unified framework to measure the resource and energy efficiency of software, making it difficult for developers and researchers to compare results, replicate findings, or systematically improve energy usage.

Despite growing awareness of the environmental impact of software, the lack of standardized tools and methods has led to fragmented and inconsistent efforts in evaluating and optimizing software efficiency, hindering progress in reducing its environmental footprint. The issue of **green software** is becoming increasingly urgent. As global attention to sustainability rises, the carbon footprint of software has emerged as a critical challenge. Traditional software development often overlooks energy efficiency and carbon emissions, leading to resource waste.

While green technologies are advancing in many industries, the software sector lags behind. Without a standardized framework for green software measurement, unchecked software-driven energy consumption will continue to burden the ICT industry and slow global progress toward carbon neutrality and sustainability goals.

2 Proposed solution

Our solution focuses on designing a data processing and analysis system that prioritizes energy efficiency, making it greener. This is achieved through several key design choices at different stages of the system architecture, including data collection, storage, and front-end optimizations. Related work on green software engineering either focus on best practices but without quantifying their impact on a holistic basis or propose an evaluation methodology without concrete examples from real-world applications. By drawing from the lessons learned in the literature, our solution incorporates green software principles while providing a practical example of implementation in a real-world context. We compare and contrast two distinct system design architectures to explore and discuss the following: first, the options to consider for creating a greener system design in real-world scenarios; second, methods for quantifying the impact of greener implementations using established evaluation techniques; and finally, the specific components that may significantly influence energy consumption.

2.1 Literature Review

Previous research indicates that while practitioners are aware of and care about energy efficiency in software engineering, they often face significant barriers due to a lack of comprehensive information and tools available. Manotas et al. conducted an empirical study that highlights this challenge, revealing that developers need support to incorporate such practices into their workflows [1]. Mourão et al. conducted a systematic mapping study that categorizes and synthesizes various researches concerning sustainable software engineering, emphasizing the need for more validation studies focusing on the impacts of sustainability in the future [2]. Brunnert proposed specific green software metrics,

resource demand, as the basis to help developers allocate the energy consumption of the entire server to individual software components, yet no substantial prototype has been implemented to finalize the carbon emission calculation [3]. As web technologies continue to thrive and to be adopted, various resources are available to the community, aiming to bring awareness regarding sustainable software engineering [4]. Meanwhile, best practices identified in mobile application development adapting energy constraints on mobile devices can be effectively translate to web development. Rani et al. explored 20 mobile energy patterns that could be adapted for web and interviewed several expert web developers to challenge them. Finally, two of the ported patterns were evaluated against their antipatterns in terms of energy consumption [5]. A more detailed analysis focusing on architectural design was also within the domain of mobile development, where Singh proposed a novel RMVRVM paradigm to remove the heavylifting to the back-end to reduce unnecessary data transferred to the client devices, resulting in significant decrease in battery consumption and improvement in response time. [6]

The ITC project addresses the challenge of managing vast datasets in ways that align with the sustainability goals of Indigenous communities. This management involves handling large volumes of data, closing gaps, accurately categorizing information, and identifying significant feature relationships. Effective data processing is crucial for deriving actionable insights that facilitate informed decision-making. To minimize its environmental impact, the ITC project employs a green data analysis methodology aimed at reducing waste and conserving resources, essential for lowering the ecological footprint of its data management activities.In evaluating the greenness of their data solutions, the ITC project utilizes several key metrics: Energy efficiency involves measuring the power consumption of servers, storage, and network equipment; resource utilization assesses the effectiveness of the use of computational and storage resources. The carbon footprint is estimated by calculating the total greenhouse gas emissions from the energy consumed by the IT infrastructure. Furthermore, a lifecycle assessment provides a detailed examination of the environmental impacts at each stage of the data processing solution, offering a comprehensive perspective on sustainability[7].

2.2 Standardized Data Processing Steps

- **Requirement Analysis:** This phase ensures stakeholder is on board with the defined specifications before moving to the design phase.
- **Design:** The initial design will be moderate to accommodate changes without frequent redesigns, focusing on sustainable practices that conserve resources.
- **Implementation:** The implementation will standardize data processing across different datasets, integrating government procurement lists with attributes for Indigenous businesses, and filtering out irrelevant data. This phase also includes categorizing contracts by amount, culminating in the creation of customized charts or graphs based on user requirements.

2.2.1 Initial Data Integration

- Import Procurement Lists: Start by importing government procurement lists into our database.

- Add PSIB Attribute: Introduce a new column labeled 'PSIB'. For the attribute *indigenous_ business_ en*, set the value to 'include' for entries marked as '2. Mandatory Set-Aside' and '3. Voluntary Set-Aside'. Similarly, set the attribute *indigenous_ business_ excluding_ psib_ en* to 'include' if marked 'YES'; all others should be labeled as 'exclude'.
- Remove Unnecessary Attributes: Eliminate any attributes that are not required for further analysis to streamline the data structure.

2.2.2 Attribute Matching and Categorization:

- Create 'Indigenous Business' Attribute: Add a new attribute 'indigenous business'. Match *vendor_ name* from the first table with names from the CCIB (captured via web scraping from the CCIB website) and 'Company Legal Name' from the government's indigenous list.
- Mark Matching Names: If *vendor_ name* matches any name in the CCIB or government's indigenous list, mark it as 'yes' under 'indigenous business'; otherwise, mark it as 'no'.

2.2.3 Contract Categorization by Description:

- Refine Contracts by Domain: Based on the product descriptions in the government procurement contracts, categorize each contract into specific domains, such as technology. This will allow for focused analysis on specific areas of interest.

2.2.4 Contract Amount Categorization and Data Pruning:

- Segment Contract Amounts: Divide the contract amounts into different range categories.
- Exclude Small Contracts: Remove contracts with amounts less than $1000 to simplify the data aggregation process.

2.2.5 Data Integration for Visualization:

- Aggregate Data: Based on user requirements, integrate the data from the steps above to create charts and graphs. This involves combining the categorized data into a coherent format that is ready for visualization.

2.3 Architecture Design

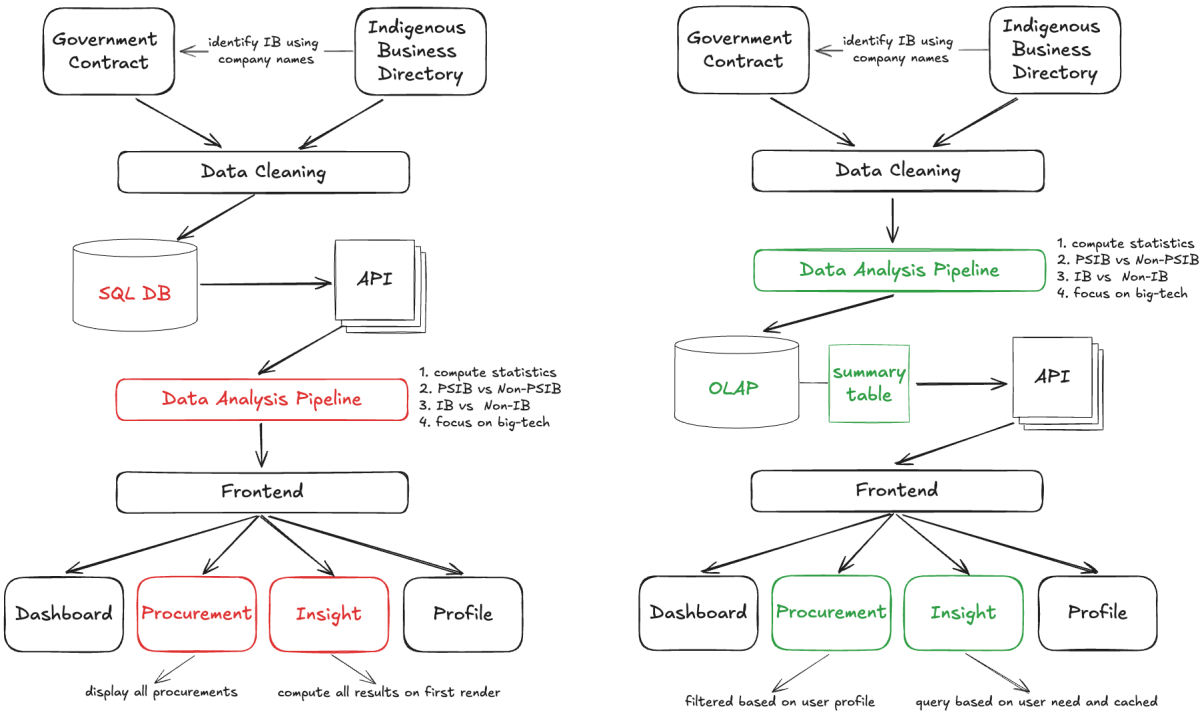

Figure 1: Two possible system architecture diagrams

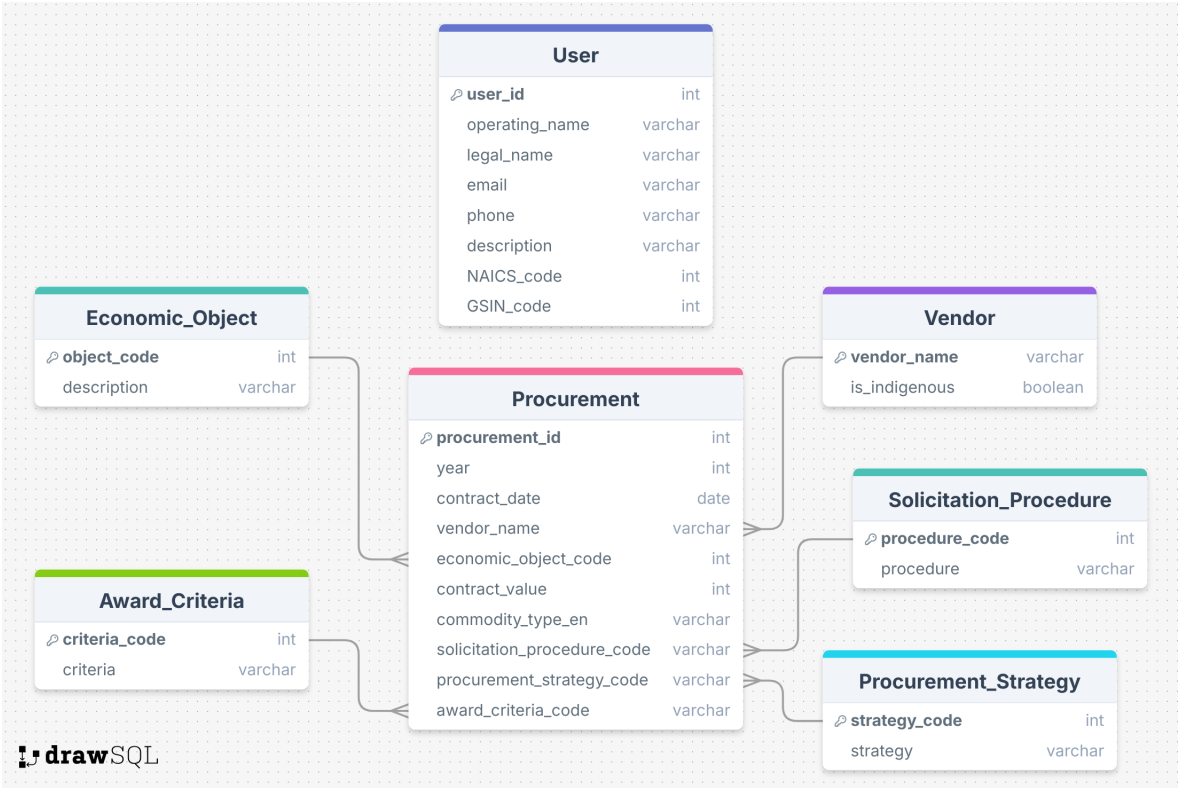

Figure 2: Relational Schema for MySQL Database

2.3.1 Data Collection

Instead of relying on real-time updates, our system will primarily conduct historical retrospective analysis. This shift is driven by the fact that the procurement data we are handling is not updated frequently. Real-time data collection can be resource-intensive, especially in terms of energy consumption and network usage.

2.3.2 Data Storage

The cleaned data will go through a data analysis pipeline and will be stored in a database. However, the type of database and the order of the process may vary. An OLAP data warehouse is preferred over a traditional relational schema because the warehouse promotes summary tables that store computed results which can be efficiently retrieved to generate charts and diagrams. Moreover, all the new data is supposed to go through the pipeline only once.

2.3.3 Front-end Optimization

While there are overwhelming amount of tips and best practices regarding sustainable web development, we decided to focus on one major aspect: allocate resources based on user needs and reduce unnecessary data flow resulted. For example, users will see procurement contracts filtered based on their company profile and they will be able to further edit it on the fly. As for the insights, data from different sections will be queried upon user's requests and rendered accordingly.

3 Evaluation Methodology

Our evaluation of software energy and resource efficiency is based on the Green Software Measurement Model (GSMM)[8]. The GSMM provides a comprehensive framework for measuring software-induced energy consumption, drawing from various international methods and approaches. We supplemented this approach with insights from the Green Coding Methodology, particularly in defining sustainable software practices

3.1 Plan for Simulation

We simulate the software under three distinct scenarios to capture a broad spectrum of performance data:

- **Idle State:** In this scenario, the system is active but no user tasks are being performed. This allows us to record the baseline energy consumption and track hardware resource usage, such as CPU and memory, when the system is not actively processing tasks.
- **Task Execution:** In this scenario, we execute typical tasks or workloads that the software is expected to handle. We record the energy consumption, CPU utilization, memory usage, and network traffic during task execution. This scenario provides an understanding of the software's efficiency during standard operations.

- **Load Test:** his stress test simulates high-load conditions by increasing the number of users or requests. The purpose of this scenario is to observe how the system scales under pressure and to record energy usage, CPU load, and memory consumption as the demand on the system increases.

These simulation repeated multiple times to account for variance in hardware efficiency and to ensure statistical significance in our results. Data are collected on CPU, memory usage, and energy consumption throughout these simulations

3.2 Data sources

The data for our simulation are obtained through system monitoring tools that provide real-time insights into hardware resource usage. In addition, we utilize existing benchmarks from Green coding methodology to compare our software against established efficiency standards. We also incorporate historical data from previous project iteration to measure improvements in energy consumption.

The Green Software Measurement Model(GSMM)[8] are applied as the primary framework for energy and resource efficiency measurements. The model enable structured analysis across different hardware components, ensuring that our evaluations are consistent and comparable with green software benchmarks.

3.3 Literature Review

The foundation of our evaluation methodology is derived from the Green Software Measurement Model (GSMM)[8], which synthesizes several established approaches for assessing the energy consumption of software. In addition to the GSMM, we referenced the Green Coding Methodology from Green Coding Documentation to further validate our sustainable software practices.

The key contributions from the literature include: Sustainable software design principles from the Green Coding Methodology, which emphasize reducing resource consumption across all stages of software development. Energy efficiency metrics provided by GSMM and Green Coding, which served as a reference for our measurement and benchmarking efforts.

3.4 Timeline

In our first evaluation process, we mainly focuses on how to setting up simulation environment and GSMM-based benchmarking tools

4 Representation of Results

4.1 Communication of Results

To convey complex data in an intuitive and accessible format, we will employ various visual tools such as charts and graphs. These visualizations will be tailored to highlight key aspects of the data, such as spending patterns, contract distribution by type, and participation levels of Indigenous businesses in government contracts.

4.2 Mock-ups

In this case, we set up specific evaluated scenario, in order to get insight of consumption in browser, which includes CPU, NETWORK, SCREEN, MEMORY, and DISK along with the graphs to illustrate the usage of CPU, and Network.

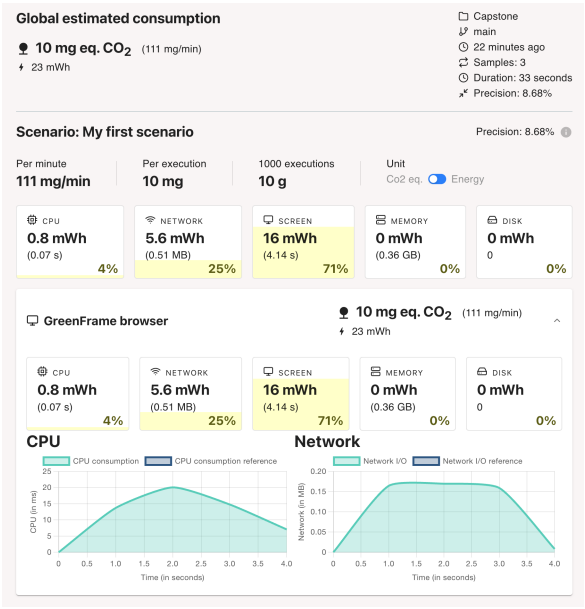

Figure 3: Evaluation of the specific scenarios with GreenFrame

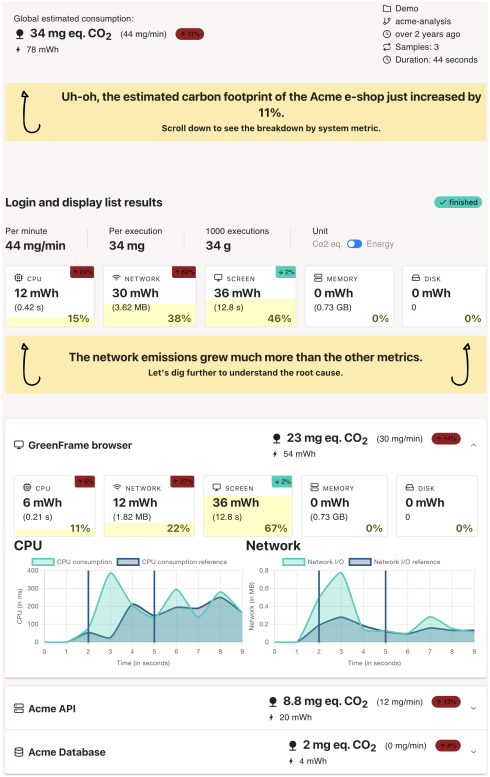

Figure 4: Demo of Evaluation with GreenFrame

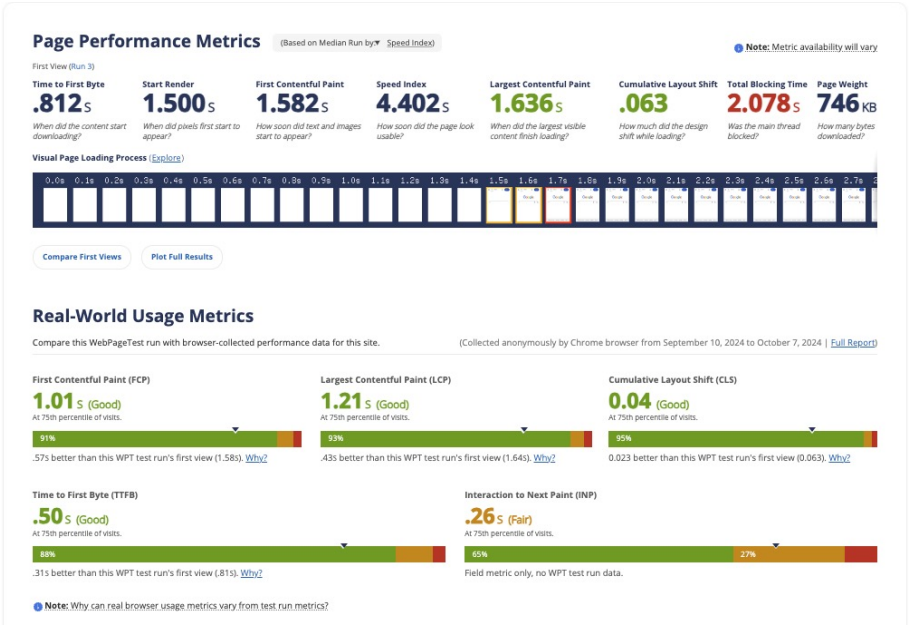

Figure 5: Demo of Evaluation with Webpagetest

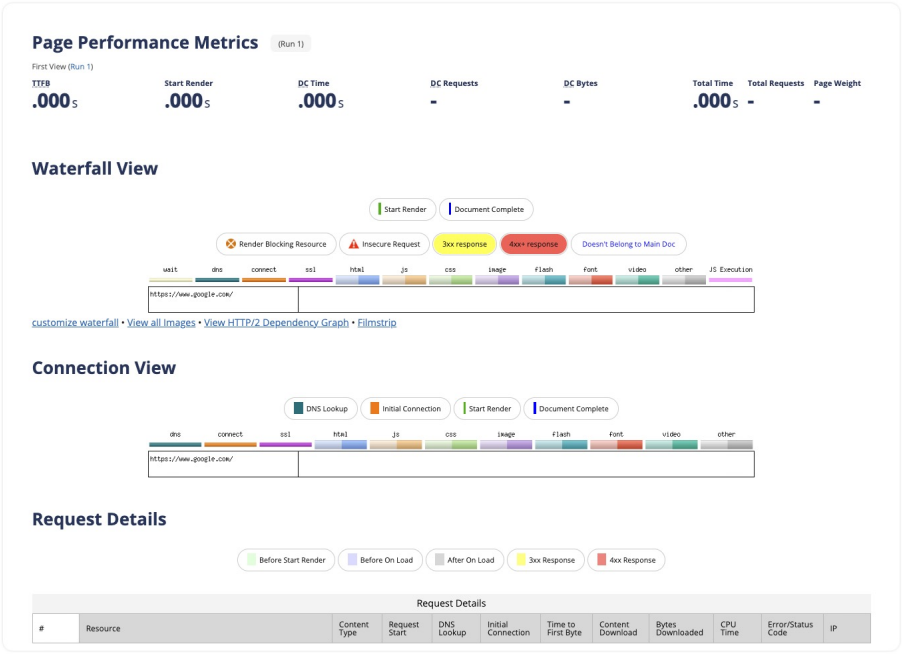

Figure 6: Demo of Evaluation with Webpagetest

5 Future work

As the field of green software engineering continues to evolve, there are several avenues for future exploration and development. First, our current solution focuses on specific aspects of data processing, front-end, and back-end optimization techniques. Future work could expand on these foundations by investigating additional areas of software architecture that impact energy efficiency. For instance, further optimization in network communications could provide deeper insights into reducing energy consumption across the entire software stack.

Another potential direction for future work involves integrating more advanced energy measurement tools. Currently, our simulations and evaluations are based on existing models such as the Green Software Measurement Model (GSMM). Further development of real-time energy tracking tools, integrated directly into development environments, could enhance the precision of energy consumption data. This would allow developers to make immediate adjustments based on feedback, enabling more dynamic optimization processes.

In addition to these technical advancements, future research should focus on the user experience within green software frameworks. While energy efficiency is a critical factor, balancing performance and usability without sacrificing environmental benefits is equally important. User-centric studies could explore how green software principles can be applied in different usage scenarios, ensuring that both energy efficiency and performance expectations are met.

Lastly, there is significant potential to broaden the scope of green software beyond individual projects to a more global perspective. Standardization efforts, particularly in the context of international software development, could help establish universally accepted frameworks and benchmarks. Collaborating with industry leaders and academic institutions on creating these global standards would accelerate the adoption of green software practices across different sectors, promoting more sustainable development worldwide.

References

[1] I. Manotas et al. "An Empirical Study of Practitioners' Perspectives on Green Software Engineering". In: *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. New York, NY, USA: Association for Computing Machinery, May 2016, pp. 237–248. DOI: 10.1145/2884781.2884810.

[2] B. C. Mourão, L. Karita, and I. do Carmo Machado. "Green and Sustainable Software Engineering - A Systematic Mapping Study". In: *Proceedings of the XVII Brazilian Symposium on Software Quality (SBQS '18)*. New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 121–130. DOI: 10.1145/3275245.3275258.

[3] A. Brunnert. "Green Software Metrics". In: *Companion of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE '24 Companion)*. New York, NY, USA: Association for Computing Machinery, May 2024, pp. 287–288. DOI: 10.1145/3629527.3652883.

[4] R. D. Caballar. "We Need to Decarbonize Software". In: *IEEE Spectrum* (Mar. 2024). Retrieved September 15, 2024. URL: https://spectrum.ieee.org/green-software.

[5] P. Rani et al. "Energy Patterns for Web: An Exploratory Study". In: *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Society*. Lisbon, Portugal: ACM, Apr. 2024, pp. 12–22. DOI: 10.1145/3639475.3640110.

[6] L. Singh. "RMVRVM – A Paradigm for Creating Energy Efficient User Applications Connected to Cloud through REST API". In: *Proceedings of the 15th Innovations in Software Engineering Conference (ISEC '22)*. New York, NY, USA: Association for Computing Machinery, Feb. 2022, pp. 1–11. DOI: 10.1145/3511430.3511434.

[7] S. R. A. Ibrahim et al. "The Development of Green Software Process Model". In: *International Journal of Advanced Computer Science and Applications* (2021). DOI: 10.14569/ijacsa.2021.0120869.

[8] Achim Guldner et al. "Development and evaluation of a reference measurement model for assessing the resource and energy efficiency of software products and components—Green Software Measurement Model (GSMM)". In: *Future Gener. Comput. Syst.* 155.C (July 2024), pp. 402–418. ISSN: 0167-739X. DOI: 10.1016/j.future.2024.01.033. URL: https://doi.org/10.1016/j.future.2024.01.033.