# Sustainable Software - GreenB

Miroslav Anatonov
Delft University of Technology
M.Sc. Computer Science
mdatanasov@tudelft.nl

Andreas Tsatsanis
Delft University of Technology
M.Sc. Computer Science
atsatsanis@tudelft.nl

Raluca Alexia Neatu
Delft University of Technology
M.Sc. Computer Science
rneatu@tudelft.nl

Francisco Duque De Morais Amaro
Delft University of Technology
M.Sc. Data Science and Artificial Intelligence
fduquedemorais@tudelft.nl

Francisco Thomas van der Boon
Delft University of Technology
M.Sc. Data Science and Artificial Intelligence
thomasboon@tudelft.nl

***Abstract*—Software running on personal devices accounts for a significant share of global electricity consumption, yet users receive little actionable feedback about which applications are responsible. This paper presents GreenB, a proof-of-concept energy monitoring system for macOS consisting of a lightweight background daemon (`greend`) and a web-based dashboard. While `greend` continuously samples per-process metrics via `powermetrics` and persists them to a local SQLite database, the dashboard surfaces historical energy data in user-friendly terms, including estimated monetary cost. A usability evaluation with ten participants found that the system communicates energy consumption effectively to non-technical users, where the main identified issues are presentational rather than conceptual, and with cost-based representation identified as the most impactful design decision. Participants also indicated a willingness to change software habits based on the information presented. Key limitations of the current prototype, mainly its proportional energy attribution model and its macOS-only compatibility, are discussed alongside directions for future work.**

## 1 Introduction

The widespread adoption of electronic devices has made them an integral part of everyday life in households around the world, with laptops and personal computers being among the most commonly owned ones. Both these devices and the software running on them collectively account for a significant and growing share of global energy consumption. However, for most users, this cost is mostly invisible: there is no persistent, interpretable record of which applications are draining their battery or increasing their electricity bill, and no mechanism by which that information might influence the software they choose to use or how developers choose to build it.

Existing operating system tools partially surface this information. Both macOS Activity Monitor and Windows Task Manager expose per-process resource metrics, but as we discuss in Section 2, their energy figures are dimensionless heuristics with no physical unit, offer no historical record, and are usually designed for one-time technical diagnostics rather than sustained user awareness. Therefore, a user is unable to learn whether their video conferencing application has consumed more energy this week than the last, or compare the efficiency of two browsers across a month of typical use.

To address this gap, we present **GreenB**, a proof-of-concept (PoC) system for continuous, per-application energy monitoring on macOS. The system aggregates this information and presents it to the user in a clear and interpretable manner, designed to be accessible to anyone regardless of their technical background. At the same time, it provides more detailed insight for advanced users and developers, allowing for a deeper analysis of software-level energy consumption. This work makes the following contributions:

- `greend`: a lightweight background daemon, written in Rust, that continuously samples per-process energy metrics from the macOS `powermetrics` utility and persists them to a local SQLite database, using dynamic idle-detection to adapt its sampling rate to system activity and minimise its own overhead.
- **A web-based dashboard**: a Flask application that reads the collected data, attributes resource usage to user-facing applications, presents energy consumption in accessible real-world terms including estimated monetary cost, and exports raw data as CSV for further analysis.
- **A small-scale usability evaluation**: a Usability Heuristics study conducted with ten participants, evaluating the dashboard's effectiveness at communicating energy information to users with varying levels of technical background.

The remainder of this paper is organised as follows. Section 2 surveys the landscape of software energy measurement, OS monitoring utilities, and related tools. Section 3 describes the design and implementation of both `greend` and the dashboard, and outlines the evaluation methodology. Section 4 presents our results. Section 5 discusses limitations and directions for future work, and Section 6 concludes.

## 2 Background & Related Work

Before starting on the implementation of our PoC, we began by surveying the domain of software energy monitoring. The findings summarised in this section are interesting on their own, as they uncovered many misconceptions both we and other surveyed developers had. We begin by examining

what energy consumption means for software, then examine specifics of how it is measured, and lastly how this information can be accessed by us, the users.

## 2.1 Software Energy Consumption

The global information and communications technology (ICT) sector is estimated to account for between 4% and 6% of all electricity produced worldwide in 2020, a share that is projected to grow substantially over the coming decade [1]. An increasingly concerning subdivision of this are data centers, which alone accounted for approximately 1.5% of global electricity consumption in 2024, growing at around 12% per year, and are projected by the International Energy Agency to nearly double their draw by 2030 [2]. However, unlike the hardware and network infrastructure that have attracted most of this attention, the energy implications of individual software applications running on end-user devices (laptops and desktop computers) remain relatively underexplored, despite user devices accounting for the majority of use-stage ICT emissions [1]. In addition to electricity usage, personal devices have a large carbon footprint for their disposal. Li-ion battery degradation, which accelerates with both charge cycles and sustained high-temperature operation caused by heavy workloads, further motivates the examining of per-application energy use [3].

The responsibility for measuring and controlling energy use is shared across the hardware and software stack. At the lowest level, hardware sensors expose power data that only the operating system kernel can read reliably. The kernel in turn exposes a subset of this information to user-space processes through system-call interfaces and virtual file systems. For this reason, developers have to depend entirely on OS-provided abstractions when reasoning about energy consumption.

In order to determine which components of the available information is actually needed to model software energy consumption, we begin by modelling the instantaneous power consumption of a personal computer as the sum of its active subsystems, as in [4]:

$$P_{\text{system}} = P_{\text{CPU}} + P_{\text{GPU}} + P_{\text{RAM}} + P_{\text{Disk}} + P_{\text{NIC}} + P_{\text{Cool}} + P_{\text{Aux}} \quad (1)$$

where each component contributes a *static* (idle) baseline and a *dynamic* (load-dependent) portion. In a personal computer, the relative weights of these components differ substantially from server environments: display and battery-management circuitry are significant additional consumers, while cooling overhead is smaller than the ≈50% share observed in data-centre hardware [5].

Each subsystem's dynamic power varies with workload in a distinct way:

- **CPU:** executing instructions switches transistors, drawing current proportional to clock frequency, supply voltage, and **the type of instruction**. León-Vega et al. demonstrate that vector-logic and vector-arithmetic instructions consume substantially more power per cycle than scalar or branching instructions, even at the same reported CPU utilisation level [4]. This means that the OS-reported utilisation percentage is an incomplete proxy for actual CPU power: two applications both showing 50% CPU usage may draw significantly different power depending on their instruction mix. Idle wakeups (events that pull a core out of a deep low-power C-state[1]) further dissipate energy independent of useful work performed [6].
- **GPU:** rendering and compute workloads impose a variable additional load. As with the CPU, the instruction mix matters: León-Vega et al. find that scalar arithmetic on CUDA cores is the most power-intensive GPU workload, while Tensor Core operations (vector arithmetic) are comparatively more energy-efficient [4]. However, on laptops with integrated graphics, GPU power is drawn from the same package[2] power budget as the CPU.
- **Memory (DRAM):** refresh cycles[3] impose a continuous baseline cost, while memory bandwidth adds a dynamic component. Vector and memory-intensive instructions are shown by León-Vega et al. to increase CPU package[2] power substantially through their effect on the memory hierarchy [4].
- **Disk I/O:** read and write operations activate SSD flash circuitry, therefore frequent small writes are disproportionately expensive relative to sequential transfers.
- **Network:** transmitting and receiving data drives the Wi-Fi or Ethernet interface, with energy cost broadly proportional to bytes transferred.

A direct consequence of this additive power consumption model is that attributing a process's share of total system power purely from its CPU time percentage, as both `powermetrics`' Energy Impact score and most OS utilities do, introduces systematic error. Nevertheless, proportional allocation from hardware-level power counters remains the most practical approach available without hardware isolation, and is also the methodology adopted by `greend` (Section 3).

## 2.2 Measuring Energy Consumption

Hardware-level energy measurement typically relies on dedicated processor interfaces. Intel's Running Average Power Limit (RAPL) interface [7] exposes cumulative energy counters for CPU packages[2], DRAM, and integrated graphics via model-specific registers (MSRs). On Linux, these counters are accessible from user space through the `/sys/class/powercap/intel-rapl` hierarchy; on macOS, Apple exposes analogous sensor data for Apple Silicon CPU, GPU, and Apple Neural Engine (ANE) subsystems through `powermetrics`. Khan et al. evaluated RAPL's accuracy and overhead, concluding that its estimates are reliable for *relative* comparisons between workloads, while absolute values can deviate from direct external measurements by a small margin [8].

[1] C-states are processor idle states; deeper states save more power but require longer wake-up latency.

[2] *Package* refers to the collection of CPU cores on a single physical processor chip.

[3] https://en.wikipedia.org/wiki/Memory_refresh

There are, however, fundamental limits to what software-level energy accounting can capture. This attribution ambiguity begins with shared hardware resources: a process that is nominally idle but preventing the system from entering a deep sleep state cannot be straightforwardly charged for that opportunity cost. Another example, *network-induced energy* on remote infrastructure, such as routers, switches, and data-centre servers serving content to the local device, is invisible to local measurement tools entirely.

## 2.3 OS Utilities

Since our goal is to create an energy monitoring tool, we begin by surveying the existing utilities for each (major) operating system.

### 2.3.1 Linux `top` and `powertop`

The `top` utility, available on all major Linux distributions, provides a real-time view of per-process CPU and memory utilisation but offers no energy or power information. Intel's `powertop` [9] extends this by reading RAPL counters and correlating them with wakeup events per process, offering estimated power contributions. However, `powertop` is primarily an interactive diagnostic and tuning tool: it does not persist data for later analysis, and its terminal interface is unsuitable for non-technical users, or long-term studies.

### 2.3.2 Activity Monitor and `powermetrics`

On macOS, Activity Monitor is the primary graphical system monitor available to users. Its *Energy Impact* column is derived from `powermetrics`, a privileged command-line utility bundled with macOS that reads hardware power sensor data and per-process statistics at configurable intervals [10]. Energy Impact is a proprietary, dimensionless composite of CPU and GPU time, disk I/O, and networking, weighted by some coefficients that are not publicly documented. It is therefore a *relative* heuristic ranking rather than an absolute physical measurement. Apple's own documentation states explicitly that `powermetrics`' power estimates are approximate and should not be used for cross-device comparisons.

### 2.3.3 Windows Task Manager

Windows Task Manager displays per-process CPU, memory, disk, network, and GPU utilisation. Moreover, since Windows 10 it also includes a *Power usage* column providing a qualitative rating from *Very Low* to *Very High*. This rating combines CPU time, GPU time, and disk activity, but similar to Energy Impact, its exact formula is undocumented and no absolute figures in watts are exposed to the user. The Windows Energy Metering Interface (EMI) [11] offers programmatic access to hardware energy counters on supported platforms, but availability is inconsistent across device families and it remains largely absent from developer tooling.

## 2.4 Related Tools and Existing Research

Several research prototypes and commercial tools have attempted to close the gap between low-level hardware counters and developer or user-facing energy visibility. Joulemeter [12] from Microsoft Research, which is no longer available for public download, inferred machine-wide and per-VM power consumption on Windows from hardware utilisation counters using a proportional allocation model similar in spirit to the one employed by `powermetrics` (and, as we will see later, `greend`). GreenMiner [13] and Trepn [14] target the Android domain, enabling controlled energy measurement for mobile applications through external power monitors and on-device battery APIs respectively.

On the side of developer awareness and behaviour change, Pinto et al. surveyed energy-aware programming practices and found that the majority of developers are largely unaware of the energy implications of their implementation choices, and that accessible tooling is a prerequisite for improvement [15]. Manotas et al. corroborated this in an empirical study of professional software engineers, observing that energy efficiency is consistently deprioritised relative to performance and correctness, chiefly due to the absence of meaningful feedback at development time [16].

These findings directly motivate the dual focus of the present work: surfacing energy data in a form meaningful to users, while also providing the per-process detail that developers need to act on it.

# 3 Methodology

**Research Question:** *How can we encourage users to become, and remain, aware of the energy consumption of the apps they use? Furthermore, how can we use this awareness to incentivise app developers to optimise their apps?*

While OS-provided tools such as Activity Monitor and Task Manager partially address the first part of this question, Section 2 identifies two key limitations that prevent them from fully answering it: their energy figures are dimensionless and non-comparable across sessions, and they offer no persistent record from which trends can be identified over time.

To provide a possible answer to our research question we designed and implemented a PoC energy monitoring application, consisting of two parts:

- A background (daemon) process `greend` that periodically takes metrics samples from the user's system, monitoring all running applications and efficiently logging their resource consumption to a database.
- A web-based dashboard that reads the collected data, and presents it in a form meaningful to non-technical users, while still remaining informative to developers.

The scope of this PoC is intentionally limited to macOS. As discussed in Section 2, energy data collection is tightly coupled to OS-specific interfaces; `greend` relies on `powermetrics`, which is exclusive to macOS. Porting to Linux or Windows would require substituting this component with the platform's equivalent interface (RAPL via `powercap` on Linux; EMI on Windows) and re-implementing the corresponding parsing and attribution logic. The methodology and design principles described here are intended to generalise, even where the implementation currently does not.

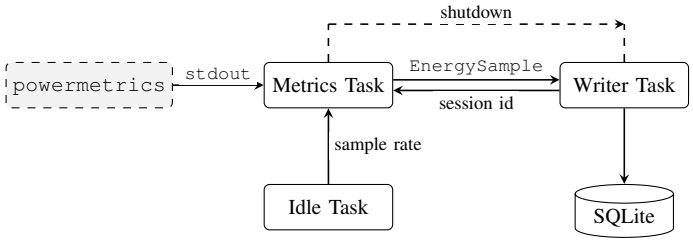


Fig. 1. `greend` daemon architecture. Solid arrows denote data flow; dashed lines denote the broadcast `watch` shutdown signal. `mpsc` carries the sample stream; `oneshot` carries the session identifier; `watch` carries sample-rate updates from the idle detector.

## 3.1 Data Collection

In order to efficiently gather usage data over long periods of a device being powered on and running, we developed a daemon[4] process `greend`. Meant to start running when the system is first booted and continue uninterrupted for the entire time the user's device is used, `greend` gathers usage metrics for all running applications, and saves them to a local SQLite database. The primary principle enforced in the development of `greend` is that it should itself use as little energy as possible. This is crucial since it would be at least ironic to have an energy monitoring application drain the user's battery.

### 3.1.1 Architecture

The daemon is structured as three concurrent asynchronous tasks running on a single-threaded Tokio runtime [17], communicating exclusively through channels (Figure 1):

- **Metrics task**: spawns `powermetrics` as a child process, continuously reads and parses its output, converts each sample into structured `EnergySample` records, and forwards them over an in-memory channel to the writer task.
- **Writer task**: receives `EnergySample` records, accumulates them in a fixed-capacity buffer, and flushes them to the SQLite database in batched transactions. It also creates a `MonitoringSession` row on startup and closes it on shutdown.
- **Idle detection task**: periodically queries the system for the time elapsed since the last user input, and dynamically adjusts the sampling interval by signalling the metrics task via a watch channel.

The choice of a single-threaded runtime avoids the overhead and memory cost of a multi-threaded executor while remaining adequate for the daemon's largely I/O-bound workload.

### 3.1.2 Metrics Source

On macOS, `greend` delegates the low-level energy measurement to `powermetrics`[5], which is invoked as a subprocess with the `plist` output format and the `--show-process-energy` flag group. This flag group enables per-process reporting of CPU time, GPU time, disk I/O, and network I/O within each sample window. `powermetrics` additionally reports the total instantaneous power draw of the CPU and GPU subsystems, in watts, derived from hardware power sensors on Apple Silicon and Intel platforms alike.

The daemon reads the streamed `powermetrics`' output, parsing each sample into a structured `PowermetricsSample`. Memory usage per process is not provided by `powermetrics` and is supplemented using `libc`, refreshing only the processes present in the current sample to minimise overhead.

### 3.1.3 Energy Attribution

`powermetrics` reports total CPU and GPU power at the package[2] level, not per process. To attribute power to individual applications, `greend` employs a proportional allocation model: each process is assigned a share of the total CPU power proportional to its share of total CPU time consumed during the sampling interval, and likewise for GPU power. Formally, for process $p$ in sample $s$:

$$P_p = P_{\text{CPU}} \cdot \frac{t_p^{\text{cpu}}}{\sum_i t_i^{\text{cpu}}} + P_{\text{GPU}} \cdot \frac{t_p^{\text{gpu}}}{\sum_i t_i^{\text{gpu}}} \tag{2}$$

where $P_{\text{CPU}}$ and $P_{\text{GPU}}$ are the total package CPU and GPU power in watts, $t_p^{\text{cpu}}$ is the CPU time of process $p$ in milliseconds per second, and the sums run over all processes in the sample. Energy consumed during the interval is then estimated as $E_p = P_p \cdot \Delta t$, where $\Delta t$ is the elapsed time of the sample window in seconds.

This model broadly adapts the approximation already present in `powermetrics`' own power estimates, which are noted by Apple to be indicative rather than precise measurements. The approach is nevertheless consistent with the methodology of `powermetrics`' built-in Energy Impact metric, and is sufficient for the comparative and awareness-raising purposes of this work.

### 3.1.4 Collected Metrics

For each process in each sample, `greend` records the following fields:

- **Power and energy:** estimated instantaneous power draw (W) and energy consumed during the interval (J).
- **CPU utilisation:** fraction of a single core's time used, expressed as a percentage.
- **GPU utilisation:** fraction of GPU time used, expressed as a percentage.
- **Memory:** resident memory usage in megabytes, sourced from the OS process table.
- **Disk I/O:** bytes read and written during the interval, converted to megabytes.
- **Network I/O:** bytes sent and received during the interval, converted to megabytes.

Each record is also associated with a session identifier, a timestamp, the process name, and its PID. Records are persisted to SQLite with indices on `session_id`, `timestamp`,

[4] A daemon is a program that runs as a background process, rather than being under the direct control of an interactive user. See https://en.wikipedia.org/wiki/Daemon_(computing)

[5] `sudo powermetrics` is a system utility bundled with macOS; https://ss64.com/mac/powermetrics.html.

and `app_name` to support efficient querying by the dashboard.

#### 3.1.5 `greend`'s Own Energy Efficiency

Since `greend` is itself a continuously running process, minimising its own energy footprint is the primary design concern. Three design decisions help achieve this goal, as we explain below.

**Dynamic sampling rate**: The idle detection task polls the system for the time since the last human input event (via `ioreg`'s `HIDIdleTime` on macOS) every 20 seconds. When the system is actively in use, `powermetrics` is invoked with a 5 second sampling interval. As the system becomes progressively idle, the interval is relaxed to 15, 30, or 60 seconds, and extended to 30 minutes when the system appears to be sleeping or unattended (idle for more than 15 minutes). This keeps the overhead of `powermetrics` itself proportionate to the relevance of the data being collected.

**Buffered database writes**: Writing to disk on every sample would incur significant I/O overhead and disk wakes. Instead, `greend` accumulates up to 2 000 samples in memory before flushing them to SQLite inside a single transaction. The database is furthermore configured with write-ahead logging[6], reducing the number of `fsync`[7] calls required.

**Single-threaded runtime**: All concurrency is expressed through async/await on a single OS thread, avoiding the overhead of thread context-switching and stack allocation that a multi-threaded design would introduce.

### 3.2 Visualisation

A key component to achieving our goal is to create clear and informative visual aids to allow for quick and accurate interpretation of our findings. As such, we decided on developing the **GreenB** dashboard, a Flask web application designed to present the collected energy data at two levels of detail: a high-level overview suitable for non-technical users, and a per-application more detailed breakdown intended for technical users and developers. Its core features and interfaces are described in the sections below. Please also note that all application screenshots were taken using generated data, and the displayed values may not reflect the real-world performance of any given application.

**Main Dashboard**: The main dashboard (Figures 2, 3 and 4) presents a system-wide summary of the most recent monitoring session. Some important figures show the total energy consumed, the equivalent estimated electricity cost, and the top energy-consuming applications ranked by total power draw. The energy consumption is intentionally also expressed in monetary terms along with the physical units (watts and joules), following the principle that familiar quantities would be easier to grasp than abstract ones for certain users. Additionally, a ranked list of applications can give the user an immediate sense of relative impact without requiring them to interpret the raw figures themselves.

**Per-Application View**: Selecting an application from the ranked list opens a detail view (Figure **??** and 6 ) showing that application's resource usage over time, broken down by CPU, GPU, memory, disk I/O, and network I/O. This view is intended for users who wish to understand *why* a given application ranks highly – for instance, distinguishing a browser that is CPU-intensive from one that is network-heavy.

**Comparison View**: The comparison dashboard (Figure 7 and 8) allows users to place two or more applications side by side across all collected metrics. This can be particularly useful for developers wishing to compare the energy footprint of different versions of their software, or for users who want to choose between two applications that offer the same functionality.

**Data Export**: The dashboard also provides an option for CSV export covering a time range specified by the user, thus enabling users or researchers to perform their own analysis on the raw sample data.

**Data Upload**: The data upload page (Figure 9) allows users to import historic CSV or JSON data into **GreenB**'s database. This enables comparisons with external data using the comparison view, or merging of different sample sets together.

### 3.3 Validation

To evaluate the dashboard's usability, we conducted a Usability Heuristics (UH) study [18] with ten participants. We selected seven of Nielsen's ten heuristics, omitting those relating to error prevention and help documentation, as these features were outside the scope of the current PoC. Table I lists the heuristics evaluated, along with their descriptions and worked examples specific to the dashboard. Each participant rated issues on a four-point severity scale (0 - no issue, 1 - cosmetic, 2 - minor, 3 - major, 4 - catastrophic).

To ensure participants evaluated the system with realistic, self-generated data rather than synthetic examples, each was asked to follow a standardised warm-up protocol before completing the survey. Participants opened three browser windows simultaneously: one streaming audio or video content, and two arranged in a split-screen configuration used for a task of their choice (such as reading, writing, or browsing) for approximately ten minutes. This protocol was implemented in order to produce consistent and comparable monitoring sessions that reflected realistic multi-application usage without imposing a scripted task that might not reflect natural behaviour. After ten minutes, participants opened the **GreenB** dashboard, viewed their session data, and completed the UH survey.

Following the survey, participants answered three structured follow-up questions designed to assess the impact of the tool on energy awareness:

[6] `PRAGMA journal_mode=WAL`, see https://en.wikipedia.org/wiki/Write-ahead_logging

[7] `fsync` synchronizes a file's in-core state with storage device https://www.man7.org/linux/man-pages/man2/fsync.2.html

- Are you energy conscious in your day-to-day life?
- Would this tool introduce some new habits or change some applications that you use?
- Are you more comfortable reading what we provided in the application, or do you prefer seeing the statistics that your current monitor shows?

# 4 Results

The implementation produced the **GreenB** dashboard described in subsection 3.2, including the main overview, per-application detail, comparison, and CSV export views shown in Appendix A. This section reports the findings of the usability evaluation conducted on the deployed system, as well as a review of the implementation performance for `greend`.

## 4.1 User Study

Ten participants completed the study following the protocol described in Section 3. Overall, the feedback confirmed that the dashboard communicates energy information effectively even to non-technical users. Moreover, the monetary cost conversion was consistently highlighted by the majority of the participants as the feature that most clearly connected energy consumption to everyday experience.

The heuristics which received the most severe ratings were *Visibility of System Status* and *Aesthetic and Minimalist Design*, both with ratings in the minor-to-major usability problem range. Participant comments centred on the font sizes being too small and insufficient contrast between text and background elements, which made some figures difficult to read. However, the issues presented were identified only as presentational rather than conceptual, which confirms that participants understood what was being shown, but found it harder than necessary to read.

Another recurring point of confusion were the physical units, with several participants reporting having no intuitive sense of scale for watts (W) and joules (J), even when values were numerically presented. This suggests that the units could be supplemented by further contextualisation – for example, expressing energy in terms of equivalent device charging times or hours of screen-on use.

Regarding the follow-up questions, the majority of participants reported not currently considering energy consumption in their day-to-day software choices. However, most of them indicated that having access to the tool would prompt at least some behaviour change, most commonly reconsidering which browser or video-conferencing application they used. When asked to compare the **GreenB** dashboard to their existing system monitor, participants consistently preferred **GreenB**'s presentation, naming the historical view and the use of familiar units as the primary advantages over the technical readout of Activity Monitor.

One complaint identified across multiple responses was the absence of filtering in the energy overview: the ranked application list includes background system processes alongside user-facing applications, which several participants found distracting. Addressing this – for example, by offering a toggle between foreground and background processes – was identified as the highest-priority usability improvement for future iterations.

## 4.2 `greend` Performance

To evaluate the energy efficiency of `greend` itself, we conducted a self-measurement study by running `greend` continuously on a single device for 48 hours under normal usage conditions. At each sampling interval, `greend` recorded its own process entry alongside all other running applications, allowing its CPU, memory, and power figures to be extracted from the collected database afterwards.

Over the observation period, `greend`'s CPU utilisation remained between $0.1\%$ and $0.3\%$, resident memory usage was stable at approximately $10\,\mathrm{MB}$, and estimated power draw averaged $1\,\mathrm{mW}$. These values are at the lower boundary of `powermetrics`' measurement resolution and therefore can carry non-trivial measurement noise; however, they still consistently place `greend` an order of magnitude below common consumer applications such as browsers and editors in all three metrics, which is sufficient to conclude that its overhead is negligible in practice.

As a secondary check, the recorded values were then cross-referenced against Activity Monitor observations taken during the same period. Since Activity Monitor derives its figures from the same `powermetrics` interface that `greend` reads, agreement between the two confirms that `greend` is not miscounting or misattributing its own resource usage.

# 5 Limitations and Future Work

The central limitation of `greend`'s energy attribution model is its use of proportional CPU and GPU time allocation as a proxy for process power consumption. As discussed in Section 2 and demonstrated by León-Vega et al. [4], OS-reported CPU utilisation is an incomplete proxy for actual power draw: two processes at the same utilisation level may consume substantially different power depending on their instructions. The proportional model adopted here, although consistent with the methodology underlying `powermetrics`' own Energy Impact metric, introduces a systematic attribution error. A more accurate approach would be to incorporate hardware performance counters (e.g. via RAPL or Apple Silicon's AMX counters), in order to weigh each process's share by instruction type, along the lines of the models proposed by León-Vega et al. Therefore, we consider this to be the most important technical improvement for future work.

On the platform side, `greend` currently targets macOS exclusively. Extending support to Linux, would substantially expand the system's applicability, and is a natural next step, since RAPL counters are accessible via the `powercap` filesystem without requiring root privileges. Similarly, Windows support (via the EMI interface) could also follow in the future.

The user study, although informative, was limited in both scale and participant diversity. Ten participants drawn from a group consisting mostly of computer science students are not representative of the general population the tool targets. A

larger and more demographically varied study would therefore be needed to draw stronger conclusions about the tool's effectiveness at raising energy awareness. Future evaluations could also include longitudinal components, in order to study whether awareness translates into sustained behaviour change over weeks of use, rather than the single-session observation conducted here.

As previously mentioned in Section 4, in terms of dashboard functionality, several concrete improvement were identified by study participants as high priority. Most notable ones are the ability to filter between foreground and background processes, larger and higher-contrast text, hover tooltips providing unit explanations, and search functionality within the application list. All of these are well-defined tasks for a future iteration.

## 6 Conclusion

Energy consumption is increasingly recognised as a main concern in software engineering, yet the gap between that recognition and actionable user-facing tooling remains wide. This work demonstrates that closing that gap is technically feasible on consumer hardware: `greend` runs continuously as an unprivileged background process with negligible overhead, and the GreenB dashboard renders months of per-application energy history in a form that non-technical users find immediately interpretable.

Perhaps the more significant finding, however, is behavioural. The majority of participants in our evaluation did not self-identify as energy-conscious prior to the study, yet most indicated a willingness to change their habits after a single ten-minute exposure to their own data. This suggests that the barrier to a more energy-aware behaviour is not motivation but visibility, and that users are willing to act, but only once the information is placed in front of them in terms they can understand. The monetary cost representation, consistently the most cited feature, underlines that grounding abstract physical units in familiar real-world quantities is not merely a design nuance but a prerequisite for genuine engagement.

More broadly, this work supports the finding of Pinto et al. [15] and Manotas et al. [16] that accessible tooling is a prerequisite for a more energy-aware behaviour, these findings carrying implications beyond this prototype. If a simple proportional attribution model with all of its known limitations is already sufficient to prompt behaviour change, then more accurate instruction-aware attribution models (such as those proposed by León-Vega et al. [4]) could unlock a much deeper level of insight, particularly for developers seeking to optimise specific workloads. Cross-platform support would extend these benefits to the much larger population of Windows and Linux users, and a longitudinal study would determine whether the awareness raised in a single session translates into durable habit change, which is the question that ultimately matters most for the environmental case this work is motivated by.

## References


[1] Parliamentary Office of Science and Technology, "Energy consumption of ICT," UK Parliament POST, Tech. Rep. PN 677, 2024. [Online]. Available: https://post.parliament.uk/research-briefings/post-pn-0677/

[2] International Energy Agency, "Energy and AI," IEA, Tech. Rep., 2024. [Online]. Available: https://www.iea.org/reports/energy-and-ai/energy-demand-from-ai

[3] S. Saxena, C. Hendricks, and M. Pecht, "Cycle life testing and modeling of graphite/licoo2 cells under different state of charge ranges," *Journal of Power Sources*, vol. 327, pp. 394–400, 2016.

[4] L. G. León-Vega, N. Tosato, and S. Cozzini, "A comprehensive analysis of process energy consumption on multi-socket systems with gpus," in *Latin American High Performance Computing Conference*. Springer, 2024, pp. 52–67.

[5] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 732–794, 2016.

[6] Intel Corporation, "Power management - technology overview," 2022. [Online]. Available: https://www.intel.com/content/www/us/en/content-details/637748/power-management-technology-overview-technology-guide.html

[7] ——, "Intel 64 and ia-32 architectures software developer's manual, volume 3b: System programming guide, part 2, chapter 14: Power and thermal management," Intel Corporation, Tech. Rep., 2016. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-3b-part-2-manual.pdf

[8] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou, "RAPL in action: Experiences in using RAPL for power measurements," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 3, no. 2, pp. 1–26, 2018.

[9] Intel Corporation, "Powertop primer," 2015. [Online]. Available: https://github.com/fenrus75/powertop

[10] S. Sheppard, "`powermetrics` macos documentation." [Online]. Available: https://ss64.com/mac/powermetrics.html

[11] Microsoft Corporation, "Energy metering interface (EMI)," 2026. [Online]. Available: https://learn.microsoft.com/en-us/windows-hardware/drivers/powermeter/energy-meter-interface

[12] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, ser. SoCC '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 39–50. [Online]. Available: https://doi.org/10.1145/1807128.1807136

[13] G. Hecht, O. Benomar, R. Rouvoy, N. Moha, and L. Duchien, “Tracking the software quality of Android applications along their evolution,” in *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2015.

[14] A. Pathak, Y. C. Hu, M. Zhang, P. Bhatt, and Z. M. Mao, “Fine-grained power modeling for smartphones using system call tracing,” in *Proceedings of the 6th ACM European Conference on Computer Systems (EuroSys)*, 2011, pp. 153–168.

[15] G. Pinto, F. Castor, and Y. D. Liu, “Mining questions about software energy consumption,” in *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR)*, 2014, pp. 22–31.

[16] I. Manotas, C. Bird, R. Zhang, D. Shepherd, C. Jaspan, C. Sadowski, L. Pollock, and J. Clause, “An empirical study of practitioners’ perspectives on green software engineering,” in *Proceedings of the 38th International Conference on Software Engineering (ICSE)*, 2016, pp. 237–248.

[17] Tokio Contributors, “Tokio: An asynchronous rust runtime,” 2026, accessed April 1, 2026. [Online]. Available: https://tokio.rs

[18] J. Nielsen, “10 Usability Heuristics for User Interface Design,” Apr. 1994. [Online]. Available: 10UsabilityHeuristicsforUserInterfaceDesign

TABLE I
USABILITY HEURISTICS FOR OUR POC

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Visibility of System Status**<br><br>The design should always keep users informed about what is going on through appropriate feedback within a reasonable amount of time.<br><br>*e.g. does the dashboard clearly show energy usage in real time.* | | | | **0** No issue<br>**1** Cosmetic issue<br>**2** Minor usability problem<br>**3** Major usability problem<br>**4** Usability catastrophe |
| **Match between the System and the Real World**<br><br>The design should speak the users' language. Use words, phrases, and concepts familiar to the users, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.<br><br>*e.g. Are the energy units understandable (€ cost instead of kWh)* | | | | **0** No issue<br>**1** Cosmetic issue<br>**2** Minor usability problem<br>**3** Major usability problem<br>**4** Usability catastrophe |
| **User Control and Freedom**<br><br>Users often perform actions by mistake. They need a clearly marked 'emergency exit' to leave the unwanted action without having to go through an extended process.<br>*e.g. Can users easily navigate, reset, export or filter data* | | | | **0** No issue<br>**1** Cosmetic issue<br>**2** Minor usability problem<br>**3** Major usability problem<br>**4** Usability catastrophe |
| **Consistency and Standards**<br>Users should not have to wonder whether different worlds, situations or actions mean the same thing. Follow platform and industry conventions.<br>*e.g. Are graphs, labels, and metrics consistent across the UI* | | | | **0** No issue<br>**1** Cosmetic issue<br>**2** Minor usability problem<br>**3** Major usability problem<br>**4** Usability catastrophe |
| **Recognition rather than Recall**<br>Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design (e.g. field labels or menu items) should be visible or easily retrievable when needed.<br>*e.g. Is information easy to interpret without prior knowledge* | | | | **0** No issue<br>**1** Cosmetic issue<br>**2** Minor usability problem<br>**3** Major usability problem<br>**4** Usability catastrophe |
| **Flexibility and Efficiency of Use**<br>Shortcuts - hidden from novice users - may speed up the interaction for the expert users so that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.<br>*e.g. Can users quickly access insights (e.g. top-energy consuming apps, CSV download)* | | | | **0** No issue<br>**1** Cosmetic issue<br>**2** Minor usability problem<br>**3** Major usability problem<br>**4** Usability catastrophe |
| **Aesthetic and minimalist design**<br>Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility.<br>*e.g. Is the dashboard uncluttered and readable* | | | | **0** No issue<br>**1** Cosmetic issue<br>**2** Minor usability problem<br>**3** Major usability problem<br>**4** Usability catastrophe |

# APPENDICES

## TABLE OF CONTENTS



## APPENDIX A
## DASHBOARD USER INTERFACE


Fig. 2. Main Dashboard - Top

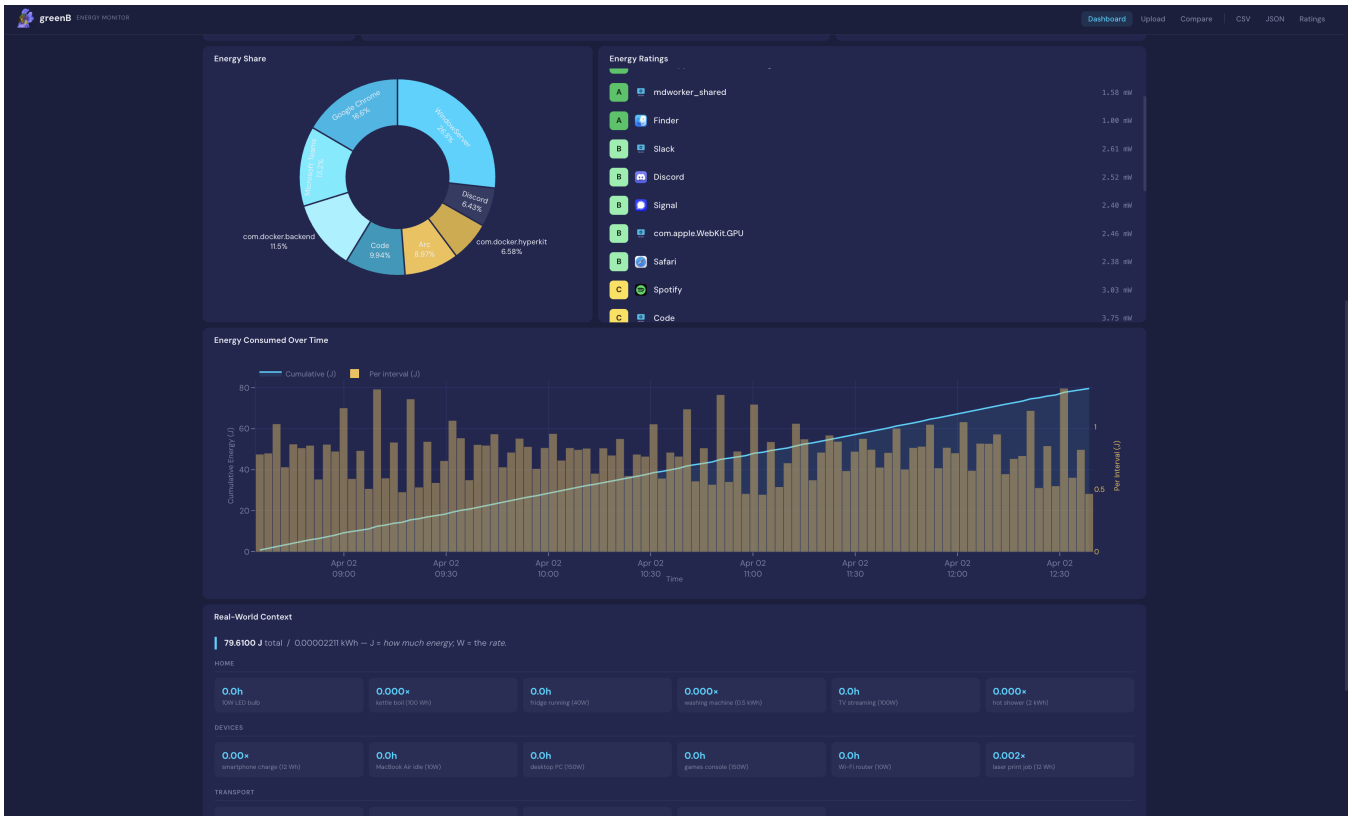


Fig. 3. Main Dashboard - Middle


Fig. 4. Main Dashboard - Bottom

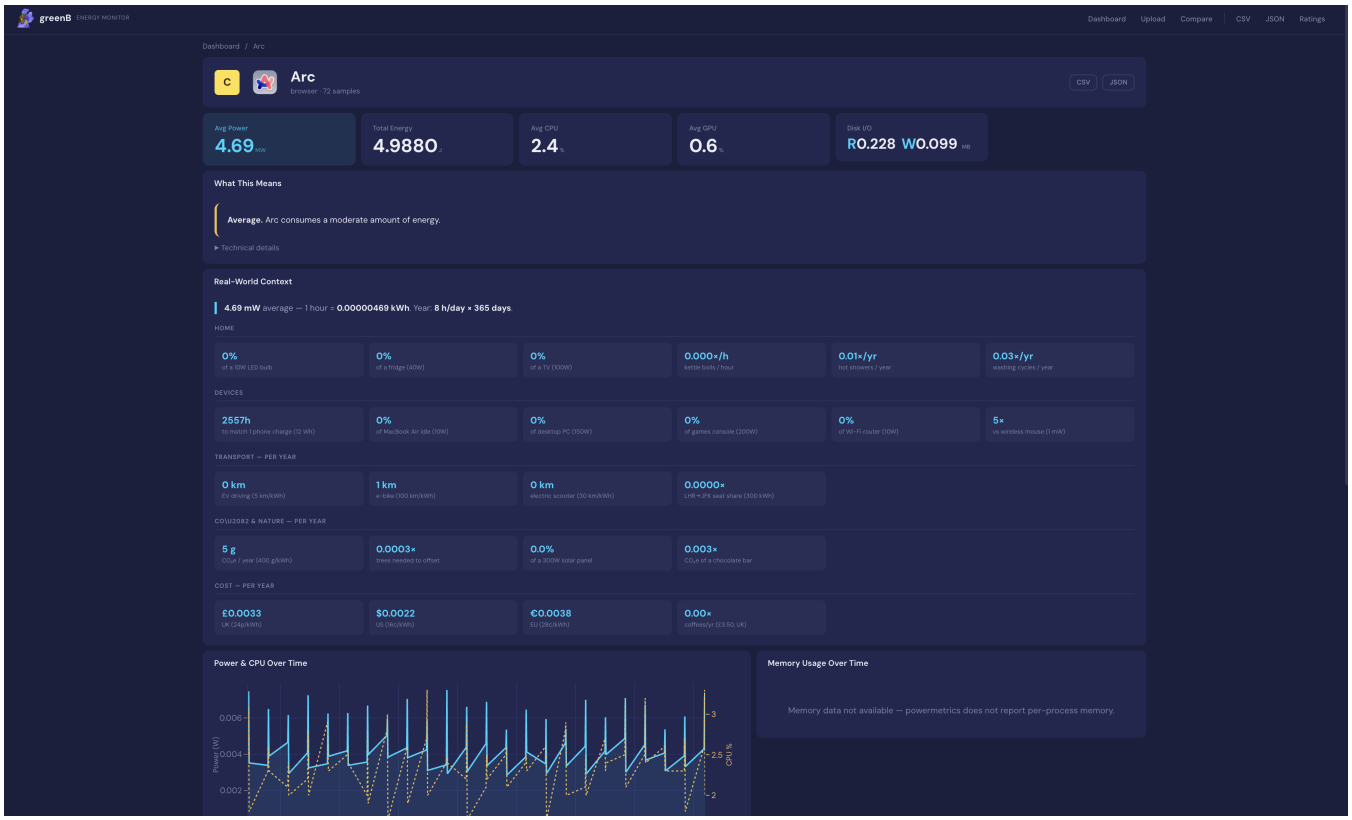


Fig. 5. Individual Application Dashboard - Top

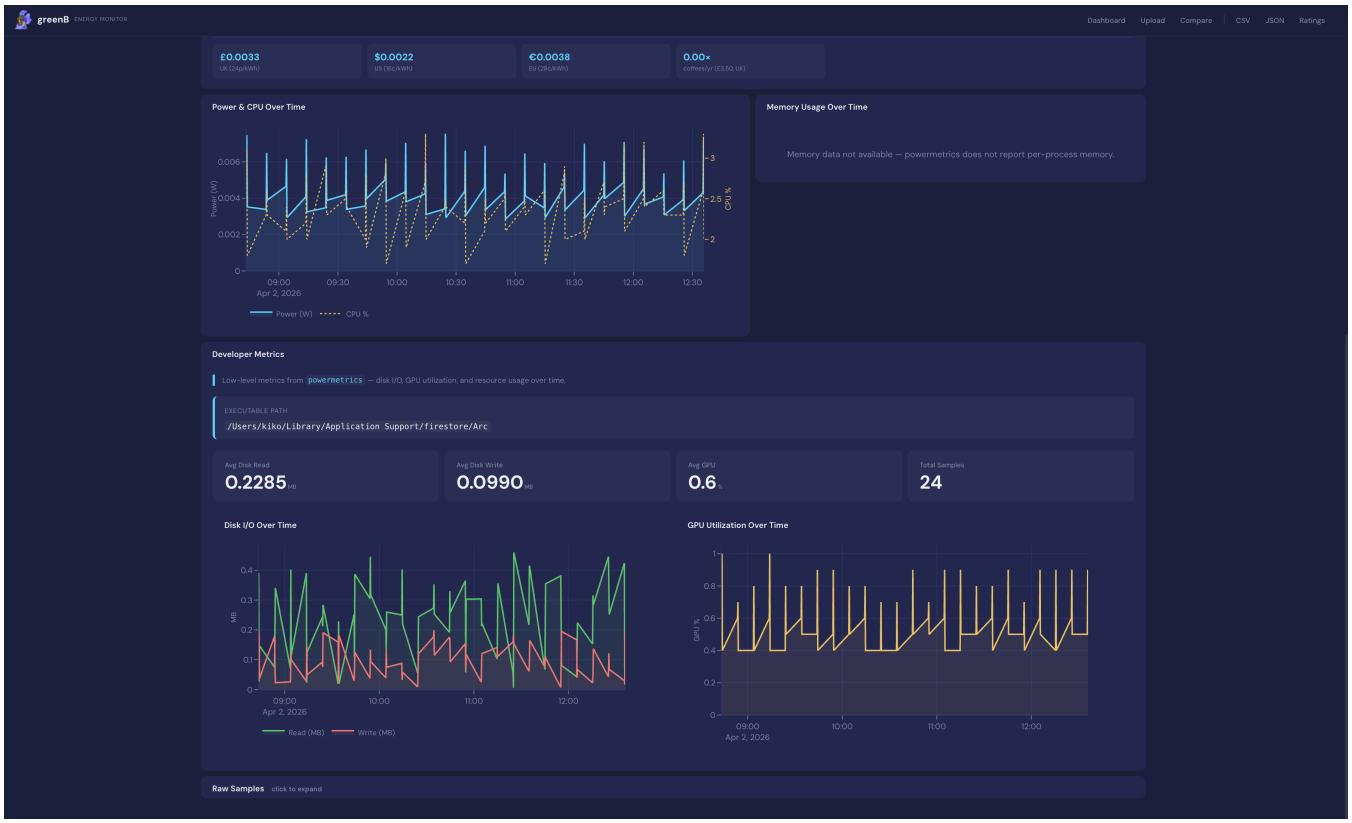


Fig. 6. Individual Application Dashboard - Bottom


Fig. 7. Inner Application Comparisons Dashboard - Top

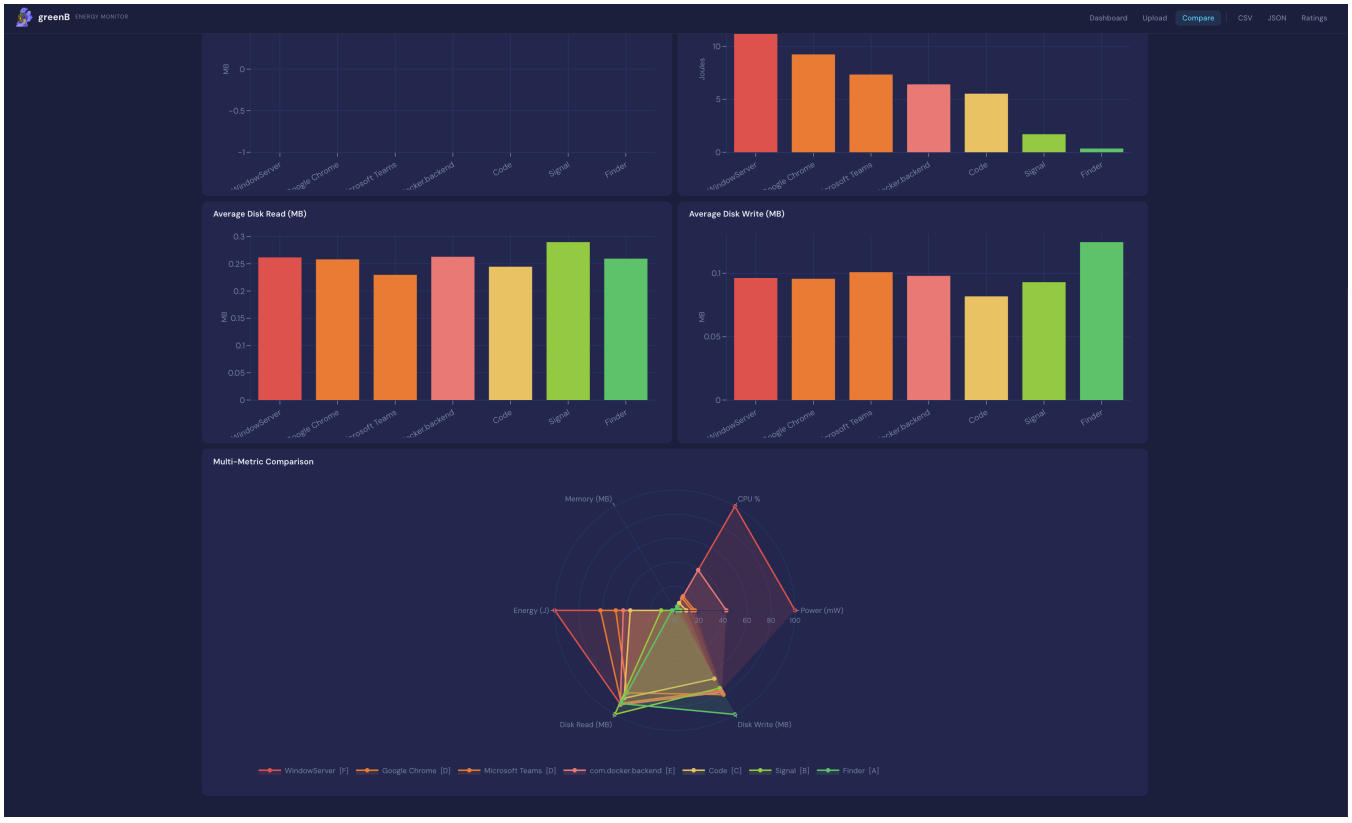


Fig. 8. Inner Application Comparisons Dashboard - Bottom

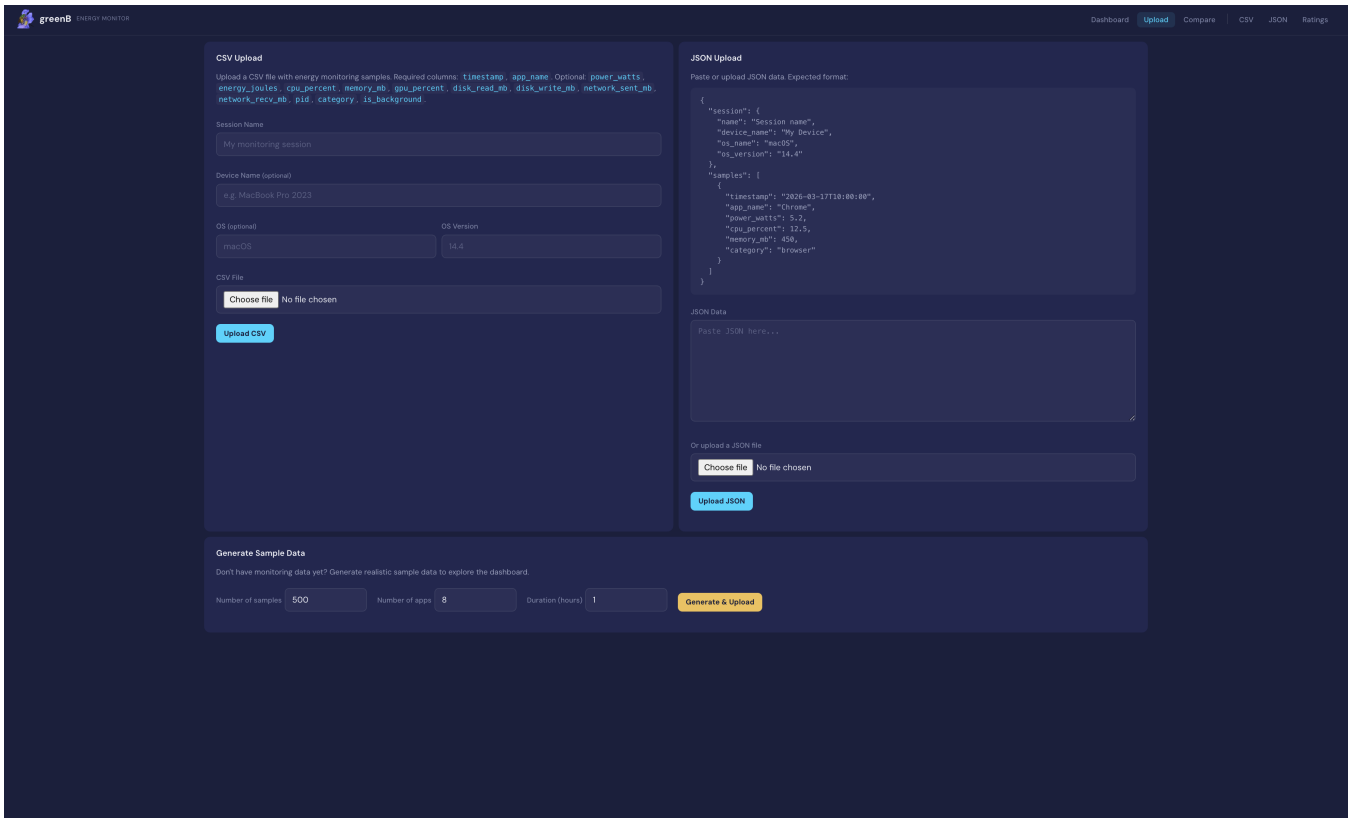


Fig. 9. Data Upload

# APPENDIX B
# USER STUDY FORMS

**Usability Heuristics Evaluation: User 1**:

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Visibility of System Status**<br>The design should always keep users informed about what is going on through appropriate feedback within a reasonable amount of time.<br>*e.g. does the dashboard clearly show energy usage in real time.* | The font size is small and difficult to read, though the information remains usable. | The overall UI could be less cluttered. The system should either use polling or Server-Sent Events (SSE) to enable live updating of the data. | | **1** Cosmetic issue |
| **Match between the System and the Real World**<br>The design should speak the users' language. Use words, phrases, and concepts familiar to the users, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.<br>*e.g. Are the energy units understandable (€ cost instead of kWh)* | No major issues were spotted. | | It is very nice to be able to see the actual monetary cost of energy usage directly in the interface. | **0** No issue |
| **User Control and Freedom**<br>Users often perform actions by mistake. They need a clearly marked 'emergency exit' to leave the unwanted action without having to go through an extended process.<br>*e.g. Can users easily navigate, reset, export or filter data* | | | The navigation and controls are very nice and easy to use. | **0** No issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Consistency and Standards**<br>Users should not have to wonder whether different words, situations or actions mean the same thing. Follow platform and industry conventions.<br>*e.g. Are graphs, labels, and metrics consistent across the UI* | | | | **0** No issue |
| **Recognition rather than Recall**<br>Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design should be visible or easily retrievable when needed.<br>*e.g. Is information easy to interpret without prior knowledge* | | | The information presented is very understandable and easy to interpret without prior knowledge. | **0** No issue |
| **Flexibility and Efficiency of Use**<br>Shortcuts — hidden from novice users — may speed up the interaction for the expert users so that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.<br>*e.g. Can users quickly access insights (e.g. top-energy consuming apps, CSV download)* | | | The interface is very easy to use and accessing key insights feels efficient. | **0** No issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Aesthetic and Minimalist Design** Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility. *e.g. Is the dashboard uncluttered and readable* | The interface feels somewhat cluttered at certain points. | | The general aesthetic is very slick and simple, which is appreciated. | **2** Minor usability problem |

*Do you consent that we use this data for our study (anonymously): yes*

**Usability Heuristics Evaluation: User 2**:

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Visibility of System Status**<br>The design should always keep users informed about what is going on through appropriate feedback within a reasonable amount of time.<br>*e.g. does the dashboard clearly show energy usage in real time.* | A live, real-time view of the data sent by the server would be preferable to the current need to fetch the newest readings. | | | **1** Cosmetic issue |
| **Match between the System and the Real World**<br>The design should speak the users' language. Use words, phrases, and concepts familiar to the users, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.<br>*e.g. Are the energy units understandable (€ cost instead of kWh)* | The user was unfamiliar with Joules (J) and Watts (W) as concepts, not knowing what these terms mean or what physical properties they measure. | | The real-world context provided in the interface is a nice touch that makes the data more easily understandable for non-technical users. | **2** Minor usability problem |
| **User Control and Freedom**<br>Users often perform actions by mistake. They need a clearly marked 'emergency exit' to leave the unwanted action without having to go through an extended process.<br>*e.g. Can users easily navigate, reset, export or filter data* | No issues identified. | | Navigation throughout the interface is easy and intuitive. | **0** No issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Consistency and Standards** Users should not have to wonder whether different words, situations or actions mean the same thing. Follow platform and industry conventions. *e.g. Are graphs, labels, and metrics consistent across the UI* | No issues identified. | | The interface feels consistent throughout. | **0** No issue |
| **Recognition rather than Recall** Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design should be visible or easily retrievable when needed. *e.g. Is information easy to interpret without prior knowledge* | The user was unfamiliar with Watts (W) and Joules (J) as terms and did not know what they measure; aside from this, the interface is easy to use. | | | **2** Minor usability problem |
| **Flexibility and Efficiency of Use** Shortcuts — hidden from novice users — may speed up the interaction for the expert users so that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. *e.g. Can users quickly access insights (e.g. top-energy consuming apps, CSV download)* | | | The efficiency features are great and well-implemented. | **0** No issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Aesthetic and Minimalist Design** Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility. *e.g. Is the dashboard uncluttered and readable* | The app selector and comparator component is very poor in its current state — it is unreadable and difficult to use. The text is too small in several places. | The app selector should be redesigned to use either a search field or dropdown menus, and could benefit from the addition of icons. | The dashboard is mainly readable outside of the problematic sections. | **3** Major usability problem |

*Do you consent that we use this data for our study (anonymously): yes*

**Usability Heuristics Evaluation: User 3**:

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Visibility of System Status**<br>The design should always keep users informed about what is going on through appropriate feedback within a reasonable amount of time.<br>*e.g. does the dashboard clearly show energy usage in real time.* | | | The display is good and the information is easily readable at a glance. | **0** No issue |
| **Match between the System and the Real World**<br>The design should speak the users' language. Use words, phrases, and concepts familiar to the users, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.<br>*e.g. Are the energy units understandable (€ cost instead of kWh)* | | | It is very helpful that real-world equivalents are provided, as this makes the data much more accessible to non-technical users. | **0** No issue |
| **User Control and Freedom**<br>Users often perform actions by mistake. They need a clearly marked 'emergency exit' to leave the unwanted action without having to go through an extended process.<br>*e.g. Can users easily navigate, reset, export or filter data* | | A download icon should be added next to the CSV and JSON export buttons to make their purpose more immediately recognisable. | Navigation is easy and intuitive overall. | **1** Cosmetic issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Consistency and Standards** Users should not have to wonder whether different words, situations or actions mean the same thing. Follow platform and industry conventions. *e.g. Are graphs, labels, and metrics consistent across the UI* | | | The interface demonstrates great consistency throughout. | **0** No issue |
| **Recognition rather than Recall** Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design should be visible or easily retrievable when needed. *e.g. Is information easy to interpret without prior knowledge* | | Some labels could be made clearer, for example by differentiating between "Compare" and "Compare Apps" more explicitly. | Recall is good overall — elements are placed where one would expect to find them. | **1** Cosmetic issue |
| **Flexibility and Efficiency of Use** Shortcuts — hidden from novice users — may speed up the interaction for the expert users so that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. *e.g. Can users quickly access insights (e.g. top-energy consuming apps, CSV download)* | There is no search bar, and the list of apps and labels is disorganised, misaligned, and poorly sectioned. Background processes are also included and there is no way to filter them out. | | | **2** Minor usability problem |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Aesthetic and Minimalist Design** Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility. *e.g. Is the dashboard uncluttered and readable* | Some areas have wasted whitespace, while others — particularly the compare section — feel overly cluttered. Additional contrast is also needed in some areas. | | The overall design is consistent, which is a strong point. | **1** Cosmetic issue |

*Do you consent that we use this data for our study (anonymously): yes*

**Usability Heuristics Evaluation: User 4**:

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Visibility of System Status**<br>The design should always keep users informed about what is going on through appropriate feedback within a reasonable amount of time.<br>*e.g. does the dashboard clearly show energy usage in real time.* | It is unclear what time period the average power metric refers to, and it is confusing that the value is lower than the figure shown for a single application. | | | **1**<br>Cosmetic issue |
| **Match between the System and the Real World**<br>The design should speak the users' language. Use words, phrases, and concepts familiar to the users, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.<br>*e.g. Are the energy units understandable (€ cost instead of kWh)* | | | The data is understandable thanks to the real-world comparisons provided alongside the technical values. | **0** No issue |
| **User Control and Freedom**<br>Users often perform actions by mistake. They need a clearly marked 'emergency exit' to leave the unwanted action without having to go through an extended process.<br>*e.g. Can users easily navigate, reset, export or filter data* | | Hover-over tooltip boxes could be added to key elements to provide contextual explanations of what each feature does. | The interface feels intuitive and easy to navigate. | **0** No issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Consistency and Standards**<br>Users should not have to wonder whether different words, situations or actions mean the same thing. Follow platform and industry conventions.<br>*e.g. Are graphs, labels, and metrics consistent across the UI* | | | The interface is consistent throughout. | **0** No issue |
| **Recognition rather than Recall**<br>Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design should be visible or easily retrievable when needed.<br>*e.g. Is information easy to interpret without prior knowledge* | | | The layout follows fairly standard conventions, making it easy to locate information without prior instruction. | **0** No issue |
| **Flexibility and Efficiency of Use**<br>Shortcuts — hidden from novice users — may speed up the interaction for the expert users so that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.<br>*e.g. Can users quickly access insights (e.g. top-energy consuming apps, CSV download)* | | | The efficiency features work fairly well and key insights are accessible without too many steps. | **0** No issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Aesthetic and Minimalist Design** Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility. *e.g. Is the dashboard uncluttered and readable* | There is some clutter in certain areas, and the font is small in places, though the overall experience is acceptable. | | | **1** Cosmetic issue |

*Do you consent that we use this data for our study (anonymously): yes*

**Usability Heuristics Evaluation: User 5**:

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Visibility of System Status**<br>The design should always keep users informed about what is going on through appropriate feedback within a reasonable amount of time.<br>*e.g. does the dashboard clearly show energy usage in real time.* | | It would be an improvement to display the exact value on top of each bar in the column chart or histogram. | | **1** Cosmetic issue |
| **Match between the System and the Real World**<br>The design should speak the users' language. Use words, phrases, and concepts familiar to the users, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.<br>*e.g. Are the energy units understandable (€ cost instead of kWh)* | | | The contextual information provided alongside the data is good and aids comprehension. | **0** No issue |
| **User Control and Freedom**<br>Users often perform actions by mistake. They need a clearly marked 'emergency exit' to leave the unwanted action without having to go through an extended process.<br>*e.g. Can users easily navigate, reset, export or filter data* | | The download button should be made more clearly identifiable, for example through the use of a recognisable icon or label. | | **2** Minor usability problem |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Consistency and Standards**<br>Users should not have to wonder whether different words, situations or actions mean the same thing. Follow platform and industry conventions.<br>*e.g. Are graphs, labels, and metrics consistent across the UI* | | | The interface is consistent in its use of terminology and visual style throughout. | **0** No issue |
| **Recognition rather than Recall**<br>Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design should be visible or easily retrievable when needed.<br>*e.g. Is information easy to interpret without prior knowledge* | | | The interface is generally intuitive and easy to navigate without requiring prior memorisation of the layout. | **0** No issue |
| **Flexibility and Efficiency of Use**<br>Shortcuts — hidden from novice users — may speed up the interaction for the expert users so that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.<br>*e.g. Can users quickly access insights (e.g. top-energy consuming apps, CSV download)* | | | Key insights are clearly presented and accessible at the top of the interface, making it efficient to use. | **0** No issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Aesthetic and Minimalist Design** Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility. *e.g. Is the dashboard uncluttered and readable* | | | The contrast between text and background is insufficient in some areas, making certain labels and informational text difficult to read. | **1** Cosmetic issue |

*Do you consent that we use this data for our study (anonymously): yes*

**Usability Heuristics Evaluation: User 6**:

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Visibility of System Status**<br>The design should always keep users informed about what is going on through appropriate feedback within a reasonable amount of time.<br>*e.g. does the dashboard clearly show energy usage in real time.* | | It would be a nice improvement to display the exact value on top of each bar in the histogram. | | **1** Cosmetic issue |
| **Match between the System and the Real World**<br>The design should speak the users' language. Use words, phrases, and concepts familiar to the users, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.<br>*e.g. Are the energy units understandable (€ cost instead of kWh)* | | | It is great that real-world equivalents are provided, as this makes the data much more accessible to non-technical users. | **0** No issue |
| **User Control and Freedom**<br>Users often perform actions by mistake. They need a clearly marked 'emergency exit' to leave the unwanted action without having to go through an extended process.<br>*e.g. Can users easily navigate, reset, export or filter data* | The download button is not clearly identifiable and its purpose is not immediately obvious. | The download button should be made more recognisable, for example through the use of a clear icon or label. | | **2** Minor usability problem |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Consistency and Standards**<br>Users should not have to wonder whether different words, situations or actions mean the same thing. Follow platform and industry conventions.<br>*e.g. Are graphs, labels, and metrics consistent across the UI* | | | The interface is consistent throughout in its use of terminology and visual style. | **0** No issue |
| **Recognition rather than Recall**<br>Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design should be visible or easily retrievable when needed.<br>*e.g. Is information easy to interpret without prior knowledge* | | Some labels could be made clearer, for example the distinction between "Compare" and "Compare Apps" is not immediately obvious. | Elements are generally placed where one would expect to find them. | **1** Cosmetic issue |
| **Flexibility and Efficiency of Use**<br>Shortcuts — hidden from novice users — may speed up the interaction for the expert users so that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.<br>*e.g. Can users quickly access insights (e.g. top-energy consuming apps, CSV download)* | | | The interface is very easy to use and key insights are accessible without too many steps. | **0** No issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Aesthetic and Minimalist Design** Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility. *e.g. Is the dashboard uncluttered and readable* | | | The contrast between text and background is insufficient in some areas, making certain labels and informational text difficult to read. | **1** Cosmetic issue |

*Do you consent that we use this data for our study (anonymously): yes*

**Usability Heuristics Evaluation: User 7**:

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Visibility of System Status**<br>The design should always keep users informed about what is going on through appropriate feedback within a reasonable amount of time.<br>*e.g. does the dashboard clearly show energy usage in real time.* | A live, real-time view of the data sent by the server would be preferable to the current need to fetch the newest readings. | | The real-world equivalent made it more clear. | **1** Cosmetic issue |
| **Match between the System and the Real World**<br>The design should speak the users' language. Use words, phrases, and concepts familiar to the users, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.<br>*e.g. Are the energy units understandable (€ cost instead of kWh)* | The user was unfamiliar with Joules (J) and Watts (W) as concepts and did not know what these terms mean or what they measure. | | | **2** Minor usability problem |
| **User Control and Freedom**<br>Users often perform actions by mistake. They need a clearly marked 'emergency exit' to leave the unwanted action without having to go through an extended process.<br>*e.g. Can users easily navigate, reset, export or filter data* | | | Navigation throughout the interface is easy and straightforward. | **0** No issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Consistency and Standards** Users should not have to wonder whether different words, situations or actions mean the same thing. Follow platform and industry conventions. *e.g. Are graphs, labels, and metrics consistent across the UI* | | | The interface feels consistent throughout. | **0** No issue |
| **Recognition rather than Recall** Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design should be visible or easily retrievable when needed. *e.g. Is information easy to interpret without prior knowledge* | | | The layout follows fairly standard conventions, making it easy to locate information without prior instruction. | **0** No issue |
| **Flexibility and Efficiency of Use** Shortcuts — hidden from novice users — may speed up the interaction for the expert users so that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. *e.g. Can users quickly access insights (e.g. top-energy consuming apps, CSV download)* | There is no search bar, and the list of apps and labels is disorganised and difficult to navigate. Background processes are also listed with no way to filter them out. | | | **2** Minor usability problem |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Aesthetic and Minimalist Design** Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility. *e.g. Is the dashboard uncluttered and readable* | The app selector and comparator is very poor in its current state — it is unreadable and the text is too small throughout. | The app selector should be redesigned using either a search field or dropdown menus, and the addition of icons would help. | | **3** Major usability problem |

*Do you consent that we use this datafor our study (anonymously): Yes*

**Usability Heuristics Evaluation: User 8**:

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Visibility of System Status**<br>The design should always keep users informed about what is going on through appropriate feedback within a reasonable amount of time.<br>*e.g. does the dashboard clearly show energy usage in real time.* | | | The display is good and the information is easily readable at a glance. | **0** No issue |
| **Match between the System and the Real World**<br>The design should speak the users' language. Use words, phrases, and concepts familiar to the users, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.<br>*e.g. Are the energy units understandable (€ cost instead of kWh)* | | | It is very nice to be able to see the actual monetary cost of energy usage directly in the interface. | **0** No issue |
| **User Control and Freedom**<br>Users often perform actions by mistake. They need a clearly marked 'emergency exit' to leave the unwanted action without having to go through an extended process.<br>*e.g. Can users easily navigate, reset, export or filter data* | | Hover-over tooltip boxes could be added to key elements to provide contextual explanations of what each feature does. | The interface is intuitive and easy to navigate. | **0** No issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Consistency and Standards**<br>Users should not have to wonder whether different words, situations or actions mean the same thing. Follow platform and industry conventions.<br>*e.g. Are graphs, labels, and metrics consistent across the UI* | | | The interface is consistent throughout. | **0** No issue |
| **Recognition rather than Recall**<br>Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design should be visible or easily retrievable when needed.<br>*e.g. Is information easy to interpret without prior knowledge* | | | The information presented is very understandable and easy to interpret without prior knowledge. | **0** No issue |
| **Flexibility and Efficiency of Use**<br>Shortcuts — hidden from novice users — may speed up the interaction for the expert users so that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.<br>*e.g. Can users quickly access insights (e.g. top-energy consuming apps, CSV download)* | | | The efficiency features work well and the interface is fairly good in this regard. | **0** No issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Aesthetic and Minimalist Design** Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility. *e.g. Is the dashboard uncluttered and readable* | The interface feels somewhat cluttered at certain points. | | The general aesthetic is slick and simple, which is a strong point. | **2** Minor usability problem |

*Do you consent that we use this data for our study (anonymously): yes*

**Usability Heuristics Evaluation: User 9**:

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Visibility of System Status**<br>The design should always keep users informed about what is going on through appropriate feedback within a reasonable amount of time.<br>*e.g. does the dashboard clearly show energy usage in real time.* | It is unclear what time period the average power metric refers to, and it is confusing that the value appears lower than the figure shown for a single application. | | | **1** Cosmetic issue |
| **Match between the System and the Real World**<br>The design should speak the users' language. Use words, phrases, and concepts familiar to the users, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.<br>*e.g. Are the energy units understandable (€ cost instead of kWh)* | | | The data is understandable thanks to the real-world comparisons provided alongside the technical values. | **0** No issue |
| **User Control and Freedom**<br>Users often perform actions by mistake. They need a clearly marked 'emergency exit' to leave the unwanted action without having to go through an extended process.<br>*e.g. Can users easily navigate, reset, export or filter data* | | A download icon should be added next to the CSV and JSON export buttons to make their purpose immediately recognisable. | | **1** Cosmetic issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Consistency and Standards** Users should not have to wonder whether different words, situations or actions mean the same thing. Follow platform and industry conventions. *e.g. Are graphs, labels, and metrics consistent across the UI* | | | The interface demonstrates great consistency throughout. | **0** No issue |
| **Recognition rather than Recall** Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design should be visible or easily retrievable when needed. *e.g. Is information easy to interpret without prior knowledge* | | | The interface is generally intuitive and elements are placed where one would expect to find them. | **0** No issue |
| **Flexibility and Efficiency of Use** Shortcuts — hidden from novice users — may speed up the interaction for the expert users so that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. *e.g. Can users quickly access insights (e.g. top-energy consuming apps, CSV download)* | | | Key insights are clearly presented and accessible at the top of the interface, making it efficient to use. | **0** No issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Aesthetic and Minimalist Design** Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility. *e.g. Is the dashboard uncluttered and readable* | Some areas have wasted whitespace while others, particularly the compare section, feel cluttered. Additional contrast is needed in some areas to improve readability. | | The overall design is consistent, which is a strong point. | **1** Cosmetic issue |

*Do you consent that we use this data for our study (anonymously): yes*

**Usability Heuristics Evaluation: User 10**:

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Visibility of System Status**<br>The design should always keep users informed about what is going on through appropriate feedback within a reasonable amount of time.<br>*e.g. does the dashboard clearly show energy usage in real time.* | The font size is small and difficult to read in places, though the information remains usable overall. | | | **1** Cosmetic issue |
| **Match between the System and the Real World**<br>The design should speak the users' language. Use words, phrases, and concepts familiar to the users, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.<br>*e.g. Are the energy units understandable (€ cost instead of kWh)* | | | The contextual information provided alongside the data is good and aids comprehension. | **0** No issue |
| **User Control and Freedom**<br>Users often perform actions by mistake. They need a clearly marked 'emergency exit' to leave the unwanted action without having to go through an extended process.<br>*e.g. Can users easily navigate, reset, export or filter data* | | | The navigation and controls are very nice and easy to use. | **0** No issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Consistency and Standards** Users should not have to wonder whether different words, situations or actions mean the same thing. Follow platform and industry conventions. *e.g. Are graphs, labels, and metrics consistent across the UI* | | | The interface is consistent in its use of terminology and visual style. | **0** No issue |
| **Recognition rather than Recall** Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design should be visible or easily retrievable when needed. *e.g. Is information easy to interpret without prior knowledge* | The user was unfamiliar with Watts (W) and Joules (J) as terms and did not know what physical properties they measure; aside from this, the interface was easy to navigate. | | | **2** Minor usability problem |
| **Flexibility and Efficiency of Use** Shortcuts — hidden from novice users — may speed up the interaction for the expert users so that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. *e.g. Can users quickly access insights (e.g. top-energy consuming apps, CSV download)* | | | The efficiency features are great and well-implemented. | **0** No issue |

| Usability Heuristic | Issues | Recommendations | Feedback | Severity |
|---|---|---|---|---|
| **Aesthetic and Minimalist Design** Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility. *e.g. Is the dashboard uncluttered and readable* | There is some clutter in certain areas and the font is small in places, though the overall experience is still acceptable. | | | **1** Cosmetic issue |

*Do you consent that we use this data for our study (anonymously): yes*

# APPENDIX C
# USER STUDY FOLLOWUP QUESTIONS

## FOLLOW-UP QUESTIONS

**User 1**

**Q: Are you energy conscious in your day-to-day life?**
A: Fairly, I try to turn things off when I'm done but I don't obsess over it.

**Q: Would this tool introduce some new habits or change some applications that you use?**
A: Probably yeah, seeing the cost would make me think twice about stuff I leave running.

**Q: Are you more comfortable reading what we provided in the application, or do you prefer seeing the statistics that your current monitor shows?**
A: I prefer this honestly, I never really looked at the monitor before.

**User 2**

**Q: Are you energy conscious in your day-to-day life?**
A: Not really, I only notice when the electricity bill comes.

**Q: Would this tool introduce some new habits or change some applications that you use?**
A: Not in this state. The app selector alone would make me close it immediately.

**Q: Are you more comfortable reading what we provided in the application, or do you prefer seeing the statistics that your current monitor shows?**
A: My monitor is fine honestly. This needs to be cleaner before I'd switch.

**User 3**

**Q: Are you energy conscious in your day-to-day life?**
A: Yeah, I try to make sustainable choices where I can, it matters to me.

**Q: Would this tool introduce some new habits or change some applications that you use?**
A: I think so, especially the comparison feature once the app list gets sorted.

**Q: Are you more comfortable reading what we provided in the application, or do you prefer seeing the statistics that your current monitor shows?**
A: Definitely this. The real world equivalents make it actually mean something.

**User 4**

**Q: Are you energy conscious in your day-to-day life?**
A: I care more about the cost than the environment side of it, but yeah I keep an eye on it.

**Q: Would this tool introduce some new habits or change some applications that you use?**
A: If the numbers are right then yes, I'd start closing things I normally leave open.

**Q: Are you more comfortable reading what we provided in the application, or do you prefer seeing the statistics that your current monitor shows?**
A: This one, the cost column is exactly what I want to see.

**User 5**

**Q: Are you energy conscious in your day-to-day life?**
A: Honestly no, it's never really crossed my mind.

**Q: Would this tool introduce some new habits or change some applications that you use?**
A: Maybe for a bit out of curiosity but I don't think I'd keep checking it.

**Q: Are you more comfortable reading what we provided in the application, or do you prefer seeing the statistics that your current monitor shows?**
A: I know the monitor is there but I never really open it, so this works fine for me.

**User 6**

**Q: Are you energy conscious in your day-to-day life?**
A: Yeah, it's something I think about both at home and at work.

**Q: Would this tool introduce some new habits or change some applications that you use?**
A: It would reinforce what I already do and the export is useful for tracking over time.

**Q: Are you more comfortable reading what we provided in the application, or do you prefer seeing the statistics that your current monitor shows?**
A: This is more readable and the cost framing gives me something to actually act on.

**User 7**

**Q: Are you energy conscious in your day-to-day life?**
A: No, not at all, it just doesn't cross my mind.

**Q: Would this tool introduce some new habits or change some applications that you use?**
A: No. The app list is a mess and I wouldn't bother with it as it is.

**Q: Are you more comfortable reading what we provided in the application, or do you prefer seeing the statistics that your current monitor shows?**
A: I know where to find the stats but I don't check them, so I can't really say one is better than the other.

**User 8**

**Q: Are you energy conscious in your day-to-day life?**
A: Very much so, it's something I actively think about and try to act on.

**Q: Would this tool introduce some new habits or change some applications that you use?**
A: Absolutely, I'd use it all the time and already see myself making choices based on what it shows.

**Q: Are you more comfortable reading what we provided in the application, or do you prefer seeing the statistics that your current monitor shows?**
A: This is way better, having the energy and cost together in one place is exactly what I want.

**User 9**

**Q: Are you energy conscious in your day-to-day life?**
A: A bit, though it's more about saving money than saving the planet for me.

**Q: Would this tool introduce some new habits or change some applications that you use?**
A: Yeah the cost breakdown would definitely make me think about what I keep running.

**Q: Are you more comfortable reading what we provided in the application, or do you prefer seeing the statistics that your current monitor shows?**
A: This one, the cost column makes it immediately clear what I'm actually spending.

**User 10**

**Q: Are you energy conscious in your day-to-day life?**
A: Not really, but I'm not against it. I don't think about it every day.

**Q: Would this tool introduce some new habits or change some applications that you use?**
A: Maybe for a while but I'm not sure it would stick.

**Q: Are you more comfortable reading what we provided in the application, or do you prefer seeing the statistics that your current monitor shows?**
A: I've opened the monitor maybe twice, so both are kind of the same to me.