Energy Research and Development Division

FINAL PROJECT REPORT

Software Development Report for the Power Management User Interface Study

California Energy Commission

Edmund G. Brown Jr., Governor

April 2019

PREPARED BY:

Primary Author(s):
Joy E. Pixley
Sergio Gago-Masague
Raquel Fallman
Sabine Kunrath

California Plug Load Research Center
California Institute for Telecommunications and Information Technology
University of California, Irvine, Irvine, CA 92697
949-824-9768
calplug.org

Contract Number: EPC-15-022

PREPARED FOR:
California Energy Commission

Bradley Meister
Project Manager

Virginia Lew
Office Manager
ENERGY RESEARCH AND DEVELOPMENT DIVISION

Laurie ten Hope
Deputy Director
ENERGY RESEARCH AND DEVELOPMENT DIVISION

Drew Bohan
Executive Director

ACKNOWLEDGEMENTS

The authors acknowledge important support from several sources. The California Energy Commission provided financial sponsorship, administrative support, and useful guidance. The California Institute for Telecommunications and Information Technology (Calit2), at the University of California, Irvine, provided additional financial and administrative support, led by Dr. G.P. Li. Members of the Technical Advisory Committee gave valuable feedback on the design of the software. The Office of Information Technology at UC Irvine provided assistance with testing the software application. Within Calit2, important contributions to the design, development, and testing of the software were made by Tina Chau, Keyvan Fatehi, Dr. Elizabeth Gervais, Steven Kuo, Viet Than Ly, and Dr. Gloria Mark.

PREFACE

The California Energy Commission's Energy Research and Development Division supports energy research and development programs to spur innovation in energy efficiency, renewable energy and advanced clean generation, energy-related environmental protection, energy transmission and distribution and transportation.

In 2012, the Electric Program Investment Charge (EPIC) was established by the California Public Utilities Commission to fund public investments in research to create and advance new energy solutions, foster regional innovation and bring ideas from the lab to the marketplace. The California Energy Commission and the state's three largest investor-owned utilities—Pacific Gas and Electric Company, San Diego Gas & Electric Company and Southern California Edison Company—were selected to administer the EPIC funds and advance novel technologies, tools, and strategies that provide benefits to their electric ratepayers.

The Energy Commission is committed to ensuring public participation in its research and development programs that promote greater reliability, lower costs, and increase safety for the California electric ratepayer and include:

- Providing societal benefits.
- Reducing greenhouse gas emission in the electricity sector at the lowest possible cost.
- Supporting California's loading order to meet energy needs first with energy efficiency and demand response, next with renewable energy (distributed generation and utility scale), and finally with clean, conventional electricity supply.
- Supporting low-emission vehicles and transportation.
- Providing economic development.
- Using ratepayer funds efficiently.

The Software Development Report for the Power Management User Interface Study is the final report for the project name project (Power Management User Interface project (Contract Number EPC-15-022) conducted by the California Plug Load Research Center at the University of California, Irvine. The information from this project contributes to the Energy Research and Development Division's EPIC Program.

For more information about the Energy Research and Development Division, please visit the Energy Commission's website at www.energy.ca.gov/research/ or contact the Energy Commission at 916-327-1551.

ABSTRACT

Research shows that desktop computers waste substantial energy due to being left on for long periods when not being used. The California Plug Load Research Center at the University of California, Irvine developed and tested a new software application that manages computer sleep settings, called Power Management User Interface (PMUI). The goal was to create a user interface that encourages desktop computer users to more efficiently use the automatic sleep settings already available through the operating system. This report describes the software application and its development. The design of the user interface was based upon past research on the efficacy of energy feedback and behavioral intervention programs. The design was modified in response to feedback from the Technical Advisory Committee, and from subjects in a laboratory pretest study. PMUI collects continuous data on sleep settings, the current state of the computer (off, sleep, active, or idle), and use of the program. The interface provides an easy-to-use sleep settings page, three pages of feedback reports about computer states, and a Frequently Asked Questions page. It uses messages and emoticons to encourage users to reduce their computer idle time. The data collection aspect of the software was modeled on standard software packages that monitor and record computer states (on, off, sleep) and user activity (active, idle). The software was developed using only free/open source elements. The application was built using Electron, a cross-platform desktop application framework. The application administers an event listener subprocess console application, written in C# for Windows operating systems and Swift for Mac operating systems, that runs in the background to collect the computer state data. The software performed as expected during a field test, which showed PMUI to be effective at changing computer users' behavior. Minor problems with the software were identifying and corrected during the field test. This report details limitations that must be addressed before distribution, and improvements suggested by the researchers. California taxpayers will benefit from this software because it can reduce energy use by desktop computers.

Keywords: *Computers, software, energy, efficiency, power management, desktops, sleep settings*

Please use the following citation for this report:

Pixley, Joy E., Sergio Gago Masague, Raquel Fallman, and Sabine Kunrath. 2019. *Software Development Report for the Power Management User Interface Study*. California Energy Commission. Publication Number: CEC-500-2019-XXX.

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

Page

EXECUTIVE SUMMARY

Introduction

A significant amount of plug-load energy goes into the use of computers in both residences and businesses. A study in several offices in California estimated that computers and monitors accounted for 66 percent of the office plug load demand (Moorefield, Frazer, and Bendt 2011). Although desktop computers are now outnumbered by laptop computers, the unit energy consumption is so much higher for desktops that their annual energy consumption (that is, the energy consumption per unit, multiplied by the number of units) remains higher than that of laptops. One national study estimated an annual energy consumption for residential computers at 18 TWh for desktops, with another 8.1 TWh for monitors, compared to 5.1 TWh for laptops (Urban et al. 2017). In the commercial sector, the annual energy consumption of an estimated 74 million desktop computers added up to 30 TWh in 2011 (Navigant Consulting 2013).

One way to save energy is to shift more computer usage from desktops to mobile computers. Another way is to make desktop computers more efficient. Both of these approaches are promising in the long run. The current project uses a third approach to save energy in the short-term: increasing the efficiency of the current stock of desktop computers by influencing users' behaviors toward power management.

All computers come equipped with low-power modes that can save a substantial amount of energy if used efficiently. The most effective method is to engage the automatic sleep settings, which transition computers into sleep mode after a specified delay period of idle, or inactive, time. Most computers are currently shipped with sleep settings already engaged. However, research suggests that desktop computers in the field are left on for long periods and spend fairly little time in sleep mode, indicating that automatic power management settings have been disabled (Roberson et al. 2004, Mercier and Moorefield 2011, Acker, Duarte, and Van Den Wymelenberg 2012, Barr, Harty, and Nero 2010, Bensch et al. 2010). At the same time, the majority of users report in surveys that their computers go to sleep when idle (Tiedemann, Sulyma, and Mazzi 2013, Pixley et al. 2014, Urban et al. 2017).

The disparity between users' self-reports of power management and researchers' observations of inefficient computers seems at least partially due to user confusion. Research suggests that many users don't understand power management or have trouble locating the settings (1E and Alliance to Save Energy 2009, Chetty et al. 2009, Pixley and Ross 2014). There are extensive public and private efforts encouraging computer users to employ power management settings and explaining how to do so, from detailed instructions on the Microsoft and Apple websites to educational campaigns at the state, city, and enterprise levels (for example, Picklum, Nordman, and Kresch 1999, Boston University 2018, City of Irvine 2018). However, these efforts do not appear sufficient for engaging and changing behaviors. An alternate approach is centralized IT control, in which an enterprise's IT department uses a software application or service to remotely control its employees' computers, including their power management settings. This works well for many enterprises and can save substantial energy. However, it is not appropriate in many situations, such as residential computer use and small companies with limited IT

resources. Also, even where centralized IT control is already used to facilitate updates and backups, a policy of mandatory sleep settings can face resistance from IT personnel and employees (Hackel, Plum, and Carter 2016, Pollard 2016).

In short, broad educational outreach has proven ineffective at convincing computer users to engage their sleep settings, and external control of those settings is limited. A third alternative is direct educational engagement with individual users: specifically, a software application that gives specific, actionable, and motivating feedback to users about their computers. Private enterprises are unlikely to fund the research and development of this solution as it would have low market value, given that the software encourages behaviors that individuals could do for free. While this is also true of other behavior modification programs, such as those aimed at dieting, exercising, or smoking cessation, consumers who pay for such solutions are already motivated by the outcome but find it difficult to change their behaviors. By contrast, changing computer sleep settings is quick and easy, and the challenge is to educate and motivate the users to do so. Funding this project through EPIC enabled a solution that can be made freely available, which should increase its uptake and thus the amount of energy saved, providing greater benefits for California ratepayers.

Project Purpose

The California Plug Load Research Center (CalPlug) at the University of California, Irvine developed and field tested a software application called the Power Management User Interface, or PMUI. PMUI was intended to increase computer users' awareness and understanding of sleep settings, give them feedback on how much time their computers spend idle, and encourage them to save energy by enabling their sleep settings. The results from the field test of this software would not only verify the effectiveness of this approach, but provide unique and useful data on computer users' behavior toward power management.

The ultimate goal was to produce a stand-alone software application that could be distributed freely and used on any desktop, at home or work, as a voluntary and cost-effective alternative to centralized IT control of power management.

This report details the development of the PMUI application. A separate report presents the findings of the field test of PMUI. The field test was conducted on over four hundred office desktops, and provided evidence that using PMUI significantly increased the number of computers with sleep settings engaged.

Project Approach

CalPlug designed, programmed, pretested, and field tested the PMUI software. The design phase was aided by experts in human-computer interaction. The program was tested on multiple computers, with a range of Windows and Mac operating systems. A laboratory pretest walked twenty-two subjects through the use of every feature to assess their comprehension and interpretation. Feedback on the functionality and design of the program was solicited from a Technical Advisory Committee comprised of industry and research experts. All possible revisions based on these suggestions were made before the field test stage. The software was vetted for security considerations by the university's Office of Information Technology.

The design of PMUI was based largely on similar efforts to encourage energy-saving behaviors more broadly, at homes and at work. Such programs and devices focus on giving users feedback on their energy usage, clarifying how to improve their behaviors and outcomes, and encouraging and motivating such changes. Research suggests several features that help promote behavior change, including frequent, specific feedback that is clear and easy to understand, engaging graphical representations, and the ability to compare one's current outcomes to prior outcomes, or to a goal or standard. These factors were incorporated into the design of the PMUI user interface.

The user interface incorporates five pages or screens which can be accessed through a side navigation pane: sleep settings, three reports on the user's computer use to date, and a help page. The focus is on encouraging users to reduce their computer's idle time: that is, the amount of time their computer is on when it is not being actively used, and could instead be in a low-power mode such as sleep.

- The *Sleep Settings* page allows changes to computer sleep and display sleep settings, presented in an easy to understand format. This page also offers the option of temporarily disabling sleep settings.
- The *Usage Report* page presents a pie chart showing the user's computer's states (off, sleep, active, and idle) and compares it to a pie chart labeled "Energy Saver Target Profile." Emoticons (happy or sad faces) give feedback on how much time the user's computer spent idle compared to the target profile, and hover text messages offer tips for improvement. The pie chart can be shown at the level of day, week, or month; navigational arrows allow the user to move back and forth between time periods. Environmental messages on this page appeal to users' values to promote energy saving.
- The *Time Spent Idle* page presents a bar chart comparing idle time for a given day of the week to the same day of the previous week. Again, emoticons are used to encourage reduced idle time, and navigational arrows allow the user to look at previous weeks.
- The *Patterns Over Time* page presents two graphs: an interval graph showing the pattern of computer states (off, sleep, active, and idle) over a certain period and a line graph showing the computer sleep setting (either "never" or the delay time setting) over the same period. The graphs can be shown at the level of month, week, day, half-day, and hour. This page helps users see how their sleep settings affect their computers' idle time, and also helps identify and trouble-shoot problems with computers that do not transition to sleep when they should.
- The *Frequently Asked Questions* page offers answers to questions about using the PMUI software and also about energy saving in computer and monitors more generally.

PMUI also includes a weekly reminder pop-up that encourages users to check their weekly Usage Report every Monday morning. The user can disable this reminder using a check-box, accessed through the Frequently Asked Questions answer about the pop-up message.

The software was designed to look and function similarly on both Windows and Mac operating systems (Windows versions 7, 8, and 10, and Mac versions from OS X Yosemite to macOS Sierra). PMUI was developed using only free or open source programming components, so that the final version can be more freely distributed and adapted.

PMUI requires an intensive data collection framework. It collects and processes time-stamped data on computer states and sleep settings to produce the user interface reports described above. It transmits that data to CalPlug's secure server, along with data on the use and functioning of the interface, and on the processes that may be preventing the computer from going to sleep if sleep is enabled.

Project Results

CalPlug field tested the PMUI software application on over four hundred office desktop computers, from March 2017 to May 2018. PMUI was installed on each subject's computer for a minimum of three months. No security issues or other negative effects were linked to the PMUI software.

Three problems were identified during the field test. The first involved subjects who logged out of their user profile at night; since the initial version of the program was installed on the profile, this resulted in missing data (shaded gray in the user interface reports). The second was that the weekly reminder pop-up appeared for only a few seconds and disappeared before most users could notice it and click on it. The third involved sleep periods showing up as idle periods, due to idle and sleep events posting at the same time. These problems were solved with updated versions of PMUI during the field test.

Other than these issues, the software functioned as expected in the field. The field test results, which are reported in detail elsewhere, show that PMUI had the hypothesized effect on behavior. Subjects who used the PMUI software were significantly more likely to enable their computer sleep settings or to reduce the delay time for their settings than subjects in the control group. This provides valuable evidence that the features incorporated in the PMUI design are effective for encouraging energy-saving behavior changes.

CalPlug has identified four modifications the PMUI software requires before being distributed widely. First, the study version of the software was designed to handle short-term data collection; a commercial version would require implementing optimized data storage and retrieval methods to accommodate long-term use. Second, the software currently cannot record sleep for computers with very short sleep delay times (less than 5 minutes). In addition, the software must be updated to reflect any subsequent upgrades in operating system or other software, and the research module that transmits data to the CalPlug server must be removed before any non-research usage.

Based on feedback from subjects, the Technical Advisory Committee, and the researchers, CalPlug also recommends several improvements to the software:

- Program the weekly reminders to be dynamic, responding to the current settings, to reduce the risk of annoying users.

- Rethink the design of the Time Spent Idle page, which often gives negative feedback even when overall usage is efficient.
- Add predictive feedback that would estimate how much idle time would change with different sleep settings, based on the user's previous behavior patterns.
- Rethink the presentation of environmental impact rewards to more effectively incentivize energy-efficient users.
- Add actionable information about troubleshooting and solving problems with computers that do not consistently transition to sleep even when sleep settings are engaged.

The current version of PMUI is designed for desktop computers. The same interface could be adapted for laptop computers with minor modifications to the sleep settings page, to distinguish sleep settings while plugged in versus while on battery power. However, user behavior for laptop computers is expected to differ enough from that of desktop computers that additional field testing would be recommended to estimate energy savings.

Finally, like any other software, distribution of PMUI to individuals requires an agent committed to maintaining the software and monitoring it for standard security updates, which is beyond the scope of CalPlug's mission.

Technology Transfer

The PMUI app was developed using open source (free) software, to better facilitate distribution to a wide range of users. As mentioned above, it has been modified to be used for non-research purposes, but would benefit from additional modifications to address issues discovered during the field test. As with any software, distribution of PMUI to individuals requires an agent(s) committed to maintaining the software and monitoring it for standard security updates.

The beta version of the standalone PMUI application is available for free on its own page on the Calit2 website (pmui.calit2.uci.edu). The availability of the app will be communicated through Calit2's extensive contact network and beyond, through the planned press release and *Interface* article. CalPlug is also in discussions with the Office of Information Technology and the Office of Sustainability about the possibility of distributing the software more broadly across campus, whether in research mode or standalone mode.

For the near-term, CalPlug will maintain and monitor the software for security and functionality and will seek independent resources to further improve the app based on feedback and problems discovered during the field test, and update the publicly available version.

CalPlug will also make the software available to other groups who want to use it as-is or to further develop it on their own. CalPlug has been demonstrating the software and its capabilities to multiple audiences, including visitors to CalPlug and the annual CalPlug Workshop and by presenting research results at conferences. Many expressed provisional interest in the program. CalPlug will continue to reach out to multiple potential agents, including government, university, utility, and commercial enterprises. The software and all relevant code will be made available to any such enterprise who wishes to adopt the technology; they may then modify the software to customize it for their own usage.

For the longer term, CalPlug plans to reach out to Microsoft, Apple, and computer manufacturers, with the goal of incorporating PMUI prior to distribution. It is assumed that this will be more effective once the software has been distributed more widely and broad interest has been established. It should be noted that even if one or both companies adopt the app, the baseline software remains open source and can continue to be used, updated, and customized by others.

Other approaches to contacting and engaging potential partners may also be explored.

Benefits to California

The PMUI software was designed to be used on any personal desktop, including residential and business. It was developed using free or open source software to facilitate its use and distribution. The goal is to distribute the software freely to the public, after taking the required steps to stabilize the software listed above (and perhaps also the recommended steps to improve the software).

The PMUI software was designed to be used on any personal desktop, including residential and commercial, expanding its potential reach to California ratepayers. The PMUI software can benefit California ratepayers both directly and indirectly. Ratepayers who use the software will learn more about power management and how to efficiently use their computers, and depending on their current settings, could save energy and money. All ratepayers will indirectly benefit from the energy that is saved by others who use the program, through reduced carbon dioxide emissions and reduced demand on the energy grid. Estimates of the effect of using PMUI are detailed in the Final Project Report, which presents the results of the field test (Pixley et al. 2019). At a projected full deployment of ten percent of California desktops, PMUI would save 16,245 MWh/year and over $2.6 million in electricity costs, and would reduce carbon dioxide emissions by 11,488 metric tons.

This research sets the groundwork for additional projects, applying the same behavioral feedback intervention strategies to laptops and to other plug load devices in homes and offices.

CHAPTER 1: Introduction

Desktop computers are responsible for a significant proportion of energy wasted in homes and enterprises. They have the highest unit energy consumption of any household electronic device, in part because of the amount of time they spend idle (Urban et al. 2017): that is, they spend long periods on but not being actively used, when they could be in a low-power mode instead. Computers come equipped with power management settings that reduce idle time if they are enabled. However, research repeatedly finds that in both homes and offices, desktop computer sleep settings are not effectively utilized (Barr, Harty, and Nero 2010, Mercier and Moorefield 2011, Acker, Duarte, and Van Den Wymelenberg 2012, Bensch et al. 2010).

At the same time, most computer users report believing that their computers are going to sleep (Tiedemann, Sulyma, and Mazzi 2013, Pixley et al. 2014, Urban et al. 2017). Some research suggests that users are confused about their power management settings, in part because they cannot tell whether their computers are asleep when their displays go to sleep (Pixley and Ross 2014) or they think that the display screen saver indicates that their computers and display are sleeping (Moorefield, Frazer, and Bendt 2011).

If confusion is the problem, effective feedback may be the solution.

To address this issue, the California Plug Load Research Center (CalPlug) at the University of California, Irvine developed and tested a software application intended to increase computer users' understanding of sleep settings and of how much time their computers spend idle, and to encourage them to save energy by reducing idle time. This report outlines the development of that software application, the Power Management User Interface, or PMUI.

CHAPTER 2: Background

Computer Energy Usage

A significant amount of plug-load energy goes into the use of computers in both residences and businesses. The annual consumption of both residential and commercial desktop computers added up to 52 TWh in 2011, second only to televisions among miscellaneous electric loads (Navigant Consulting 2013). In the commercial sector, the annual energy consumption of an estimated 74 million desktop computers added up to 30 TWh in 2011 (Navigant Consulting 2013). A study in several offices in California estimated that computers and monitors accounted for 66 percent of the office plug load (Moorefield, Frazer, and Bendt 2011). A more recent study of consumer electronics estimated the total number of desktop computers in United States homes to be 72 million, the number of monitors 101 million, and the number of portable computers 122 million (Urban et al. 2017). Despite laptops outnumbering desktop computers, desktop computers still use more energy overall: Table 1 shows that desktop computers in homes consumed 18 TWh annually, while portable computers only used 5.1 TWh. Desktop computers exhibited the highest unit energy consumption among household electronics (246 kWh per year, compared to 123 kWh per year for televisions). ENERGY STAR® site estimates that up to 50 dollars a year can be saved per computer if computer power management settings are enabled (ENERGY STAR 2018a).

Table 1: Energy Usage by Desktop Computers, Laptops, and Monitors in US Homes

	Units (millions)	Unit Energy Consumption (kWh/year)	Annual Energy Consumption (TWh)
Desktop Computer	72	246	18
Portable Computer	122	42	5.1
Monitor	101	80	8.1

Credit: Energy Consumption of Consumer Electronics in U.S. Homes in 2017 (Urban et al. 2017, p. 14).

Power Management

Definition

All modern desktop computers come equipped with low-power modes. An open industry standard exists for such settings and is used by most manufacturers: the Advanced Configuration and Power Interface, or ACPI (UEFI Forum 2018). In *sleep mode*, the computer reduces power to subsystems that are not needed (ACPI S3, commonly referred to as "sleep, standby, or suspend to RAM"). Random-access memory (RAM) is powered at a minimal level; open programs and files are held in RAM, allowing for a quick revival of recent activity. *Hibernation* (ACPI S4) deactivates almost all computer functions; after storing the current state of processes to the hard disk the computer powers down. Like sleep, hibernation restores any open programs and files when the computer is reactivated, but it requires more time to reactivate it. S*hutdown* or *soft off* (ACPI G2/S5) also powers down the computer but does not save the current processes and therefore all data must be saved and programs must be closed. *Hybrid* sleep is a relatively new option that combines sleep mode and hibernation. It consumes approximately as much energy as sleep, but it also saves processes to the hard drive for retrieval, which is useful if there is a power loss.

These low-power modes use substantially less energy than idle mode. Urban et al. (2017) estimated that in 2016, desktop computers on average used 56 watts in short-idle mode and 52 watts in long-idle mode, but only 2.1 watts in sleep mode and 0.9 watts when shut down. The ENERGY STAR power management calculator uses estimates of 48.1 watts for idle mode versus 2.3 watts for sleep mode for average desktop computers or, for ENERGY STAR-compliant computers, 27.1 watts versus 1.8 watts (ENERGY STAR 2018c). The challenge, then, is to ensure that desktop computers spend as little time as possible on when they are not being actively used. In theory, automatic power management settings, particularly the sleep settings for computer and display, should address this problem. The settings allow the user to control how and when various components transition automatically to various low-power modes. ENERGY STAR-labeled computers have automatic power management enabled when shipped by the manufacturer. The default settings include transitioning the computer to sleep mode after a period of 30 minutes of inactivity (Intel 2009). Computer users can also use manual power management: that is, shutting down the computer or putting it into sleep or hibernate mode immediately using menu commands.

Implementation

Past research suggests that desktop computers are left on for long periods and spend fairly little time in sleep mode, indicating that automatic power management settings have been disabled (Roberson et al. 2004, Mercier and Moorefield 2011, Acker, Duarte, and Van Den Wymelenberg 2012, Barr, Harty, and Nero 2010, Bensch et al. 2010). At the same time, the majority of users report in surveys that their computers go to sleep when idle (Tiedemann, Sulyma, and Mazzi 2013, Pixley et al. 2014, Urban et al. 2017).

A study conducted at UC Irvine shed some light on the apparent contradiction by comparing self-reports to researcher observations in the same sample. Pixley and Ross (2014) found that

while 78 percent of subjects reported that their office desktops automatically transitioned to sleep, only 20 percent actually had computer sleep settings enabled. Some of that discrepancy may be due to social desirability reporting bias. However, subjects who reported incorrectly were more likely to have not changed their settings recently (or ever) and to rate themselves as lower on computer expertise, suggesting that they were honestly confused about whether their computers were going to sleep. Also, most of these users had display sleep enabled, and may not have been able to discern whether their computers were still on when their monitors went dark. Screen savers can add to the problem, as users might confuse their screen saver with computer and/or monitor sleep. Moorefield, Frazer, and Bendt (2011) found that screen savers were used often when desktop computers were on during nonworking hours.

Other studies also suggest that users don't understand power management or have trouble locating the settings. For instance, one survey of over 1400 United States employees who use computers at work found that 32 percent either did not know what power scheme settings were (e.g., what "Balanced" or "Power saver" schemes mean), did not how to change their power management settings, or both (1E and Alliance to Save Energy 2009). An in-depth study of households in Seattle similarly found that many people they interviewed "did not know how to alter their power settings to be more energy efficient" (Chetty et al. 2009 p. 1039).

Education

There are extensive public and private efforts encouraging computer users to employ power management settings and explaining how to do so. ENERGY STAR has a set of activation instructions for different operating systems on its website (ENERGY STAR 2018b). Households and organizations can pledge to enable their power management settings and calculate their annual savings resulting from using computer power management (ENERGY STAR 2018d, 2013). Through the Federal Electronics Challenge, the federal government encouraged federal agencies to enable ENERGY STAR power management features on 100 percent of eligible computers and monitors (United States Environmental Protection Agency 2018). Many universities, cities, and companies have websites, fliers or posters encouraging the use of power management options and providing instructions (for example, Picklum, Nordman, and Kresch 1999, Boston University 2018, City of Irvine 2018). Instructions for engaging sleep settings for each specific operating system can be found on the Microsoft and Apple websites, as well as countless other IT help pages. However, given the research cited earlier that sleep settings are disabled on most desktop computers, these wide-ranging educational efforts do not appear to be effective.

Centralized Power Management

An alternate approach is centralized IT control, in which an enterprise's IT department uses a software application or service to remotely control its employees' computers, including their power management settings. This works well for many enterprises and can save substantial energy, but is not appropriate for everyone. Small companies may not be willing or able to commit the necessary resources and IT staffing. Higher-status professional employees and those with computer expertise can push back about losing control over their computers (Hackel, Plum, and Carter 2016, Pollard 2016). Some organizations may have so many

employees opting out (say, because they need to remotely access their desktops) that the savings are not worth the investment. Some IT managers resist the idea of centralized power management, expecting it to increase their work load through additional trouble-shooting, complaints, and issues with the networked environment, such as software distribution, update procedures, security patches, and backup management (Korn et al. 2004, Orland et al. 2014). Also, power management might be inconvenient to centrally manage because it is connected to the user's account and not to the computer; if the user is logged out of the account the IT manager cannot access the settings (Korn et al. 2004). Furthermore, such an approach does nothing for residential desktops or for those used by the self-employed.

When centralized IT control is not desirable or not an option, individual users should be encouraged to enable power management settings on their own computers, by increasing their knowledge, skills, and motivation.

User Feedback

As outlined earlier, research suggests that one of the reasons computer sleep settings are not enabled is that users are confused about their settings. A similar issue has been found in household energy consumption. When individuals do not understand how much electricity and gas their household uses, and on what applications, they cannot make informed decisions about saving energy (Darby 2006).

Feedback has been used to address this problem by making energy use visible and accessible by directing the users' attention to the usage (Darby 2006, Delmas, Fischlein, and Asensio 2013, Fischer 2008, Karlin, Zinger, and Ford 2015). Feedback can increase users' knowledge about energy (Ehrhardt-Martinez, Donnelly, and Laitner 2010, Lynham et al. 2016, Promann and Brunswicker 2017) and allow them to track and monitor their usage (Karlin 2011, Hermsen et al. 2016). It may instill a sense of control over energy usage and point to the relevance of the user's behavior (Darby 2006, Delmas, Fischlein, and Asensio 2013, Fischer 2008). It can motivate consumers to use less by activating motives like saving money or decreasing the environmental impact of power usage (Fischer 2008, Promann and Brunswicker 2017). Feedback can also offer users a convenient way to fulfill their energy-related goals if coupled with full or partial automatization (Promann and Brunswicker 2017) or highly personalized recommendations (Froehlich 2009).

Many types of informational feedback mechanisms have been developed, from web applications displaying utility customers' current and past monthly bills to in-home displays producing real-time feedback. Research on these feedback interfaces have identified several aspects of data presentation that tend to be more effective in engaging users' attention and encouraging them to change their behavior. Simply stated, interventions that give feedback, motivate and engage the users, and incorporate a goal with frequent well-designed messages seem to be most effective (Bird and Legault 2018). The next subsections briefly explain the elements of successful feedback and give examples of how these were incorporated in the development of the PMUI user interface.

Presentation of the Outcome Variable

Energy use should be presented in terms that are meaningful and relevant to the user (Lesic et al. 2018). Studies revealed that people do not have a good understanding of technical units like kWh. Cost can be a more natural unit users can relate to (Blasch, Filippini, and Kumar 2017, Darby 2006, Fischer 2008, Karjalainen 2011). However, even for people who prefer seeing energy expressed in monetary units, this natural understanding may not improve outcomes, either for learning about how to save energy (Krishnamurti et al. 2013) or observed energy reductions (Schultz et al. 2015). Furthermore, displaying cost may not be motivating if the computer user is not responsible for paying the electricity bill, for example, for an office computer (Yun et al. 2017). Framing the impacts of energy reduction in terms of units of carbon dioxide emissions, which are associated with climate change, can be successful in increasing conservation intentions (Spence et al. 2014). PMUI cannot access the actual energy being used by the computer, and as a standalone program, has no way of accessing the user's utility account or rates for usage. Instead, it focuses on reducing computer idle time, which the laboratory pretest showed was meaningful and easy for subjects to understand. Further, the software relates reducing idle time to improvements in environmental indicators, such as "tons fewer carbon dioxide emissions" and “acres of trees planted” as meaningful units.

Frequency of Feedback

Feedback can be provided at a range of frequencies, from plug meters showing current power consumption to summaries of monthly usage that are posted or mailed after the fact. When possible, data should be provided real-time or soon after consumption, so that users can link their behaviors to their outcomes (Roberts and Baker 2003, Abrahamse et al. 2005, Vine, Buys, and Morris 2013). PMUI shows computer states (active, idle, off, and sleep) both currently and in different temporal resolutions (hourly, half-day, daily, weekly, monthly).

Specificity of Feedback

Feedback is more useful when it provides users with specific, actionable information. For instance, feedback about energy usage for a user's apartment or dorm room is more informative than energy usage for the entire building. Likewise, if feedback is specific to certain energy uses (e.g., HVAC) or individual appliances, this helps users identify what they could do to save energy (Electric Power Research Institute 2009, Fischer 2008, Karjalainen 2011). This concept motivates personalized recommendations included in home energy audits, appliance-specific metering, and efforts to disaggregate smart meter data to fine-tune home energy reporting (Armel et al. 2013, Gupta and Chakravarty 2013). PMUI provides feedback specific to a single device and offers clear instructions for how to reduce energy use for this device.

Visual Presentation of the Data

Data should be clearly and simply presented, preferably in graphical form rather than tables, and multiple formats should be used to appeal to users with different preferences (Fischer 2008, Murugesan, Hoda, and Salcic 2015, Roberts and Baker 2003). PMUI offers multiple types of graphs for feedback reports (specifically: pie chart, bar chart, interval graph, and line graph), allowing users to check their usage from different perspectives. It also engages users with

pictorial representations (emoticons, trees, exhaust fumes, cars). The sleep settings are also displayed graphically, making them clear and easy to understand.

Historical Comparisons

Savings can be motivated by historical comparisons which display how the user's energy consumption has varied over time; for instance, this month compared to the same month last year (Roberts and Baker 2003, Darby 2006). Studies have shown that many users prefer this format, as it can reward improvements (Karjalainen 2011, Canfield, Bruine de Bruin, and Wong-Parodi 2017). PMUI allows users to compare their current and past computer states and directly connects past changes in sleep settings with corresponding changes in idle time.

Social Comparisons

Savings can also be motivated by comparisons to similar users (Jain, Taylor, and Peschiera 2012, Ferraro and Price 2013). However, this approach may backfire and produce a “boomerang effect” for those who are doing much better than others (Schultz et al. 2007): those who learn they are much more efficient than their neighbors may feel they can afford to indulge in energy-inefficient behaviors (Buchanan, Russo, and Anderson 2015, Byrne, Nauze, and Martin 2018). The PMUI application could not employ comparisons to other users due to the competing priority of making the final application capable of stand-alone use. That is, users will not need to be customers of a particular utility, members of an organization, or signed up along with their neighbors in order to use PMUI. This is intended to expand the scope of users who will be able to download and benefit from the application once it is finalized and available to the public.

Comparisons to a Standard

Goal setting is an important part of successful feedback, as feedback by itself represents rather neutral information (McCalley and Midden 2002). Individuals can use feedback to achieve goals by comparing their actual performance to an ideal outcome. Historical and social comparisons are thought to operate by implying goals: that is, by motivating users to improve compared to their own past performance or relative to similar neighbors. A goal can also be derived by comparing one’s performance to an internally or externally set standard (Kluger and DeNisi 1996, Karlin, Zinger, and Ford 2015). PMUI provides an external standard in the form of an "Energy Saver Target Profile" to which it compares the idle time of the user's computer.

Appealing to Values and Norms

Feedback is more likely to change behavior when it motivates users to change by appealing to personal values or social norms. For example, Schultz et al. (2007) found that the inclusion of an emoticon (happy or sad face) conveying social approval or disapproval, made study participants lower their consumption or, if they were already below average levels, keep it low. Delmas and Kaiser (2014) pre-tested various kinds of messages designed to activate users' values, to see which kind of information would motivate users most to conserve energy. They found that environmental messages (e.g., that higher electricity usage translated to removing trees from the community, increasing carbon dioxide emissions from coal plants, and adding cars to the road) were a top reason to reduce energy use for 70 percent of their pretest sample.

Results from intervention studies show that health-based messages and environmental messages may influence conservation behavior or the motivation to show pro-environmental behavior (Asensio and Delmas 2015, Steinhorst and Klöckner 2017). The PMUI software rewards low computer idle time with emoticons (e.g., "smiley faces") and appeals to environmental values such as reducing pollution.

User Engagement

Finally, feedback only works if users pay attention to it, and keep paying attention over time (Murtagh et al. 2013). To encourage this, feedback should be engaging and interactive (Fischer 2008, Vine, Buys, and Morris 2013). The PMUI design is interactive, allowing users to engage with their feedback data by zooming in or out to smaller or larger time frames, and back and forth through their own usage history. As mentioned earlier, multiple types of graphical representation are used to attempt to appeal to a wider range of users. Also, the interface provides hover text messages that give users additional information, rewarding them for engaging with the graphs, charts, and emoticons. Furthermore, to keep users engaged over time, a weekly pop-up reminder was built into the program, to encourage users to check their latest usage report.

Approach

When designing the first draft of the PMUI program and interface, the research team considered the research results summarized above, especially the features most likely to promote energy savings for this type of software approach. The design was presented to the Technical Advisory Committee in September 2016, and numerous improvements were made based on their suggestions. A second Technical Advisory Committee meeting was held a month later, to present the changes and solicit additional comments. The development team was not able to enact all of the Committee's suggestions due to timeline constraints, but may be able to do so in later versions. The research team conducted a laboratory pretest with twenty-two subjects in December 2016. Subjects were instructed to interact with each feature of the interface, and asked about their comprehension of and feedback about each page to assess its usability. (The pretest also assessed subjects' comprehension of selected questions designed for the surveys used in the field test.) Further revisions were made to the interface based on the pretest results.

In the next sections, the development of the software, the final design of the user interface, and the types of data collected are described in detail.

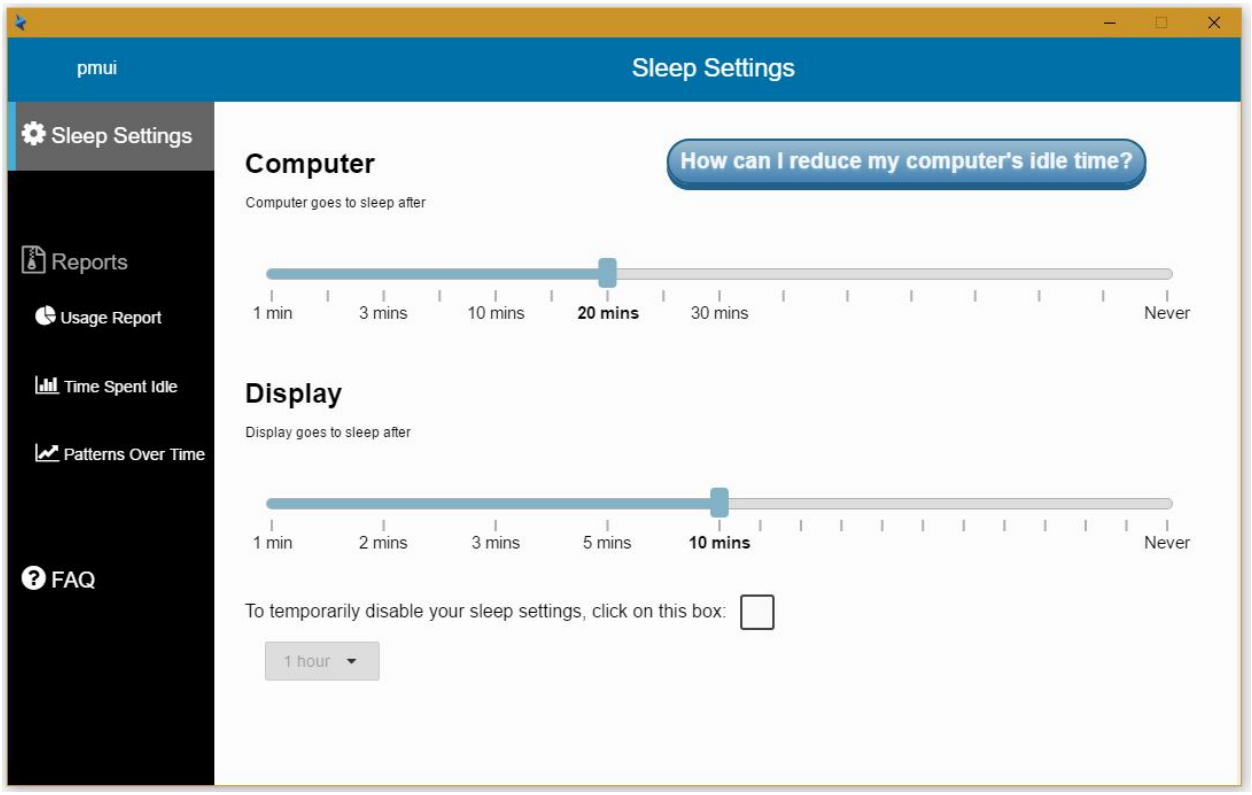
CHAPTER 3:
Design

This section describes the design of the user interface and its features. The user interface incorporates five pages or screens which can be accessed through a side navigation pane: sleep settings, a help page, and three reports on the user's computer use to date.

Sleep Settings

The *Sleep Settings* page includes three settings users can change: computer sleep settings, display (monitor) sleep settings, and an option for temporarily disabling sleep settings (see Figure 1). The computer and display sleep settings use slide bars, with the selected delay time indicated by a dark marker and a bold-face time. This design was chosen for its ease of understanding and ease of use. The program is linked to the computer system settings, so a change at either interface will appear in the other. The PMUI sleep settings include the same delay time options as the standard sleep settings on the Windows or Mac operating systems. This is necessary to maintain comparability across the two interfaces.

Figure 1: Sleep Settings Page



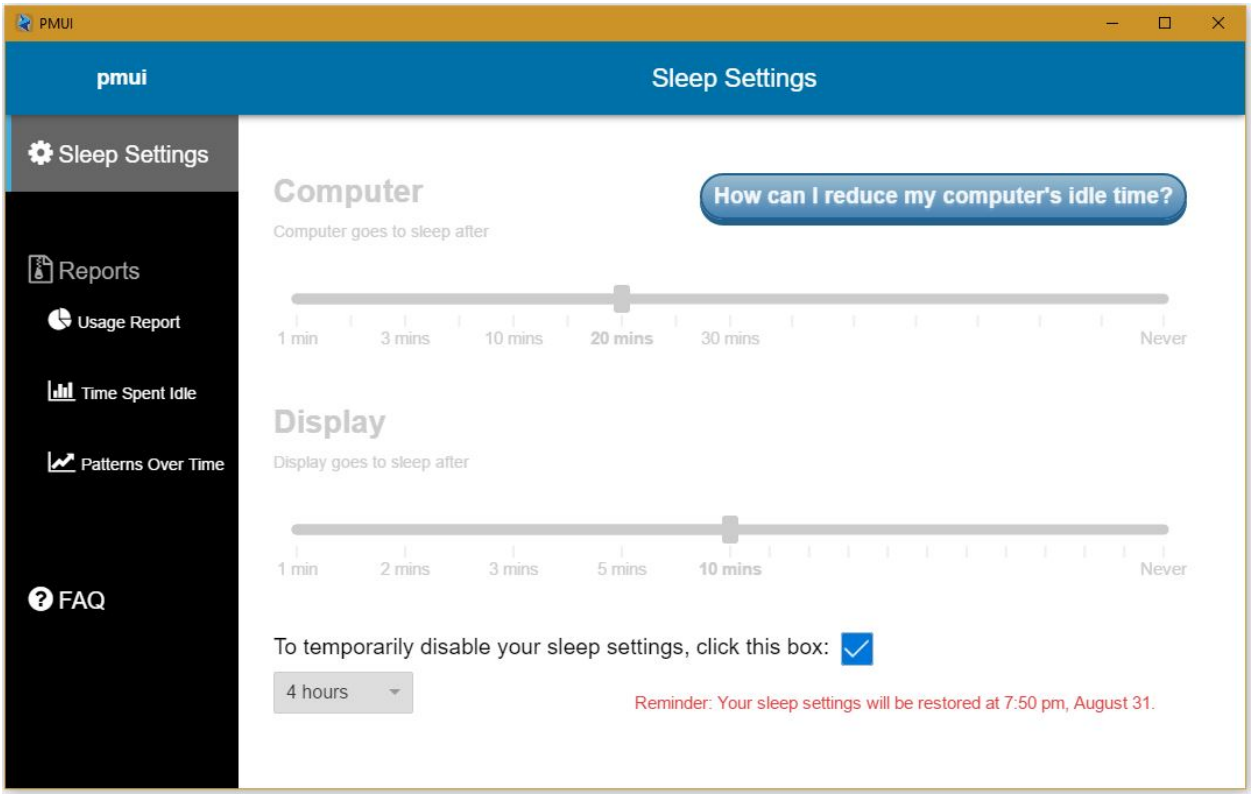
Credit: Authors

The presentation of the sleep setting options in the PMUI program is designed to communicate to the user what the "normal" or desirable settings are. The "normal" delay time settings the research team chose to promote (30 minutes for computer, 10 minutes for display) are positioned in the middle of the slide bar. The shorter delay times to the left are visible (suggesting they are preferable options) while the longer delay times to the right are not seen unless the user clicks on them. This design was based on research suggesting that people prefer to think of themselves as normal or average if not better than average, and find it aversive to be worse than average. This follows from a classic effect witnessed in survey research: subjects offered response categories for reporting socially undesirable behavior (e.g., how many hours of television do you watch per week) tend to choose the middle or lower answer, regardless of the actual range of answers, in theory because they interpret the middle category to be average (Schwarz et al. 1985). Thus, PMUI users should be subtly encouraged to choose the middle (normal) delay time or an even shorter delay time.

Different hover box texts are shown when the user moves the mouse to the button "How can I reduce my computer's idle time?" depending on the current computer sleep settings. If computer sleep settings are disabled, the message reads, "Enabling your computer sleep settings is the best way to reduce computer idle time and save energy. Tip: Try setting the computer's sleep delay to 30 minutes or less." If computer sleep is enabled but the sleep delay is higher than twenty minutes, the message reads: "Lowering the amount of time until your computer goes to sleep will reduce the time your computer is left idle, and thus save energy. Tip: Try reducing the computer's sleep delay to 20 minutes or less." If the computer's sleep delay is set at twenty minutes or lower, the same message is shown except the tip suggests "10 minutes or less."

The Sleep Settings page also allows users to disable sleep settings for a certain time period if they wish to do so. This novel feature is not provided in the native applications of Windows and Mac operating systems. The option for temporarily disabling sleep settings was included based on the notion that users may disable their sleep settings because they are worried their computer will transition to sleep mode while being used, such as while they are giving a presentation, downloading large files, or running a complex analysis. If they disable the sleep settings, they could easily forget to re-enable them later. By putting the option for temporarily disabling the sleep settings on the same page as the settings, this feature should discourage users from disabling the sleep settings in such situations. Instead, they can select a time frame (1 hour, 2 hours, 4 hours, 8 hours, 1 day, 2 day, or 3 days) and click the box. While the settings are disabled, the computer and display settings are grayed out, and a message appears notifying the user of when the settings will resume (see Figure 2). The user can unclick the check box to re-engage the sleep settings at any time.

Figure 2: Sleep Settings Page with Temporary Disable Function Activated



Credit: Authors

Reports

Three report pages were designed to give users a wide variety of feedback on their usage patterns, and to help keep them engaged and interacting with the program. The type of feedback the program can provide is limited by the standalone nature of the program: it can only report on data it collects from the computer itself and not, for instance, data on their utility company rates. The PMUI reports use data on the computer states, or more precisely, the combination of computer and user states. The PMUI program records the computer as being on, in sleep mode, or off. Whenever the computer is on, it is recorded as being actively used (mouse or keyboard activity) or idle. There are brief periods of "unknown" time when the computer turns on or wakes from sleep but cannot yet record user activity, or when the computer state cannot be determined for some other reason. In testing, these periods were usually less than one minute in duration; for the feedback reports, brief periods spent in unknown states are ignored and the prior state is extended to fill the time. Longer periods of unknown are shown in gray, and indicate the need for troubleshooting.

The reports give feedback only on observed usage of the computer, not the display. This decision was deemed reasonable because prior research by this research team showed high rates of engagement for display sleep settings. This also simplifies the interface by reducing the number of charts shown by half.

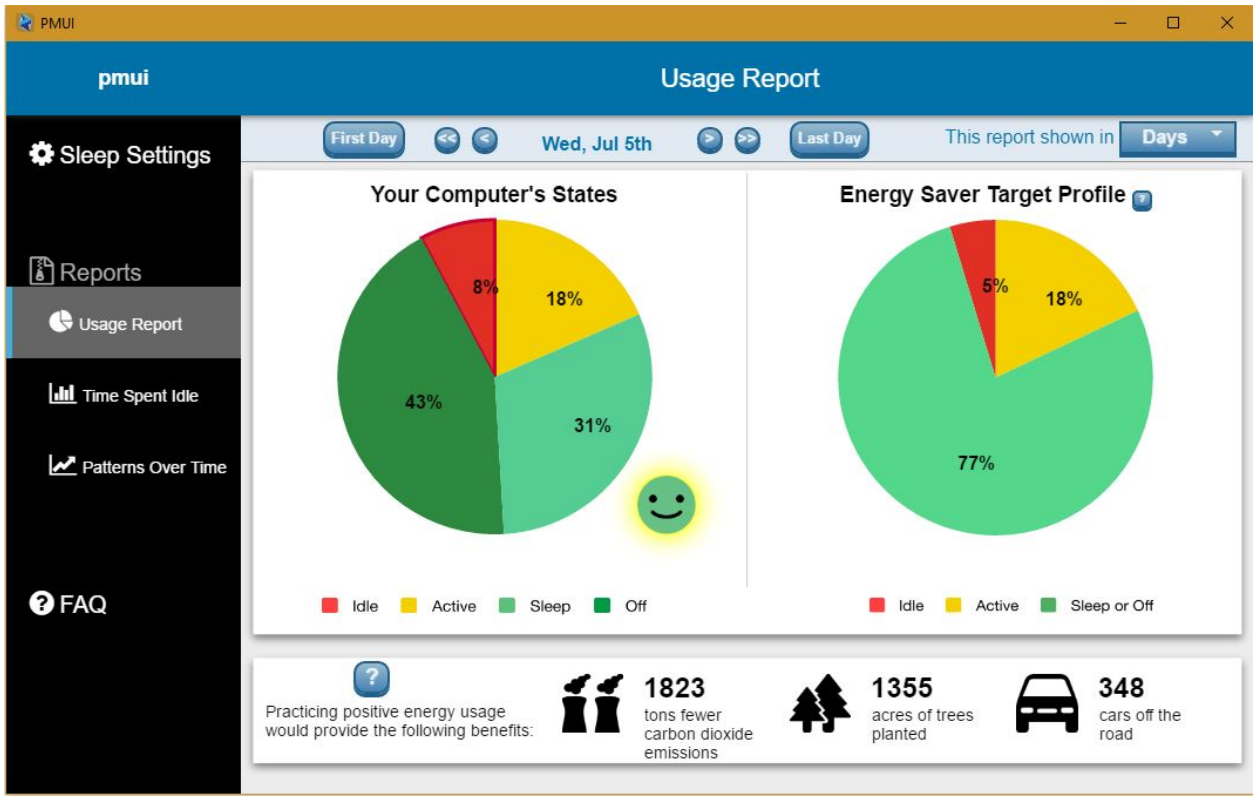
One report shows patterns on a weekly basis while the other two allow the user to change the units of time shown (e.g., days, weeks, months). All three reports allow the user to change which dates are shown, in order to scan back and forward in time to observe any changes in behavior and outcomes. These features were included to increase possibilities for user interaction with the program and allow the user to compare current and past outcomes.

This section first describes the types of graphs shown in each report page, and then some of the features that are common across the reports.

Usage Report

The *Usage Report* page shows one pie chart with the user's computer states for a given day, week, or month (see Figure 3). Specifically, it shows the percentage of time the computer spent in the four defined usage states: idle, active, sleep, and off. A second pie chart is labeled "Energy Savings Target Profile," and serves as a comparison standard. The target profile numbers were generated by simulating the percentages of idle and sleep time that would result, on average, if the computer was set to a sleep delay of 10 minutes and showed the same amount of active time as the user's profile. This target was chosen to be challenging while allowing positive feedback to be reachable for most users. The target profile pie chart combines sleep and off time in one pie wedge to avoid either advocating or penalizing the option of shutting down the computer. An emoticon appears in the left box to indicate the comparison of the user's computer idle time to that of the Energy Saver Target Profile. The rules for showing happy versus sad faces are explained in the "Motivating Energy Savings" section, below.

Figure 3: Usage Report Page



Credit: Authors

Color coding is used to suggest positive or negative valence (intrinsic goodness or badness) for the states shown in the pie charts. The attractive or desirable states (sleep, off, or either) are shown in green, which is associated with "green energy" and also with "go" as a stop light analogy. Idle time is presented as red, which is associated with "stop" in the stop light analogy, as well as with evil, red-lining, and needing correction.

A hover text box is keyed to the red "idle time" pie wedge which explains that lowering the amount of time until the computer goes to sleep will reduce idle time, and offers a link to the sleep settings page. Another hover text box appears if the user puts the cursor over the emoticon, which gives additional feedback and offers a link to the sleep settings page. These messages are explained in more detail in the next section.

A third hover text box is keyed to a question mark near the Energy Saver Target Profile and reads, "What is this?" Clicking the link takes the user to the relevant FAQ answer.

A pull-down menu allows the user to look at these pie charts by day, week, or month. As the pie chart shows a percentage, it must have data for the full period shown, so the most recent full day will be yesterday and the most recent full week will be the previous Monday through Sunday. Navigation arrows and buttons at the top of the page allow the user to move back and forth across the time period PMUI has recorded so far. For instance, in "day" mode, the user can return to the first day and then jump back to the last day, use the single arrows to move back

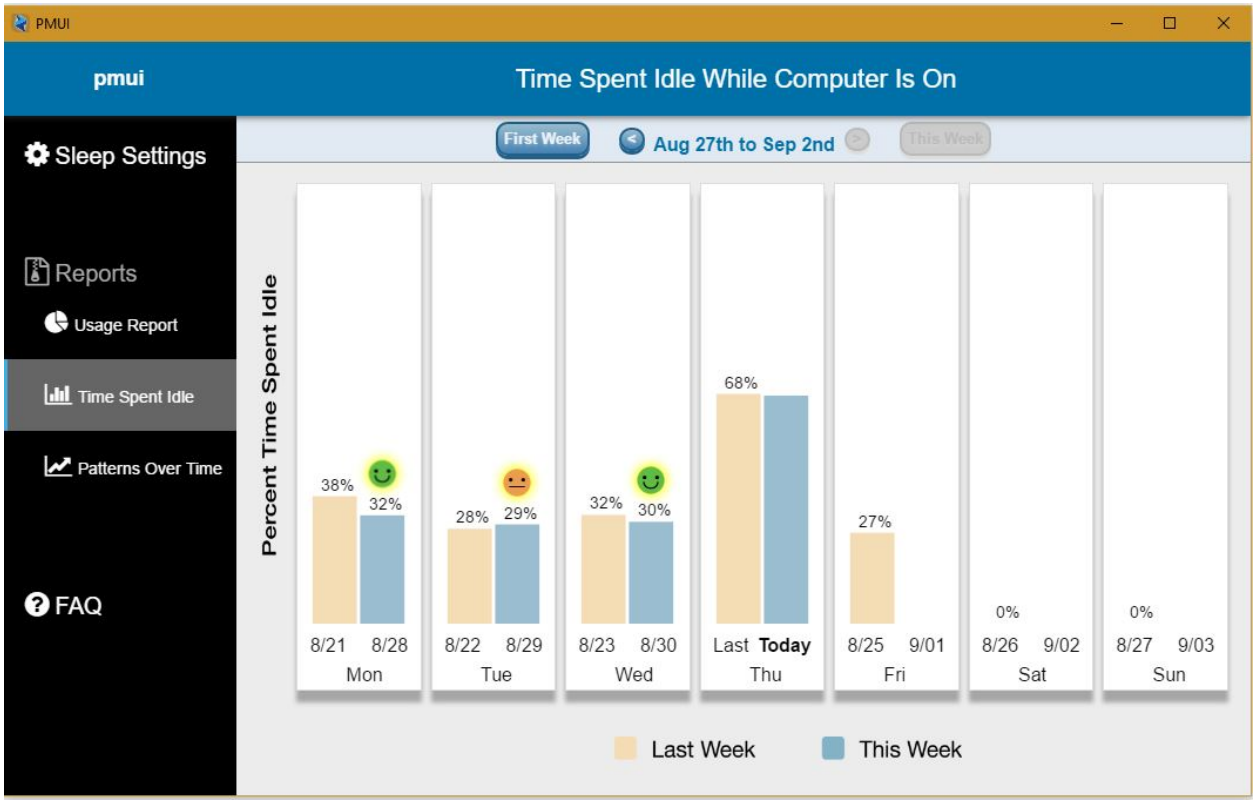
and forth by one day at a time, or use the double arrows to jump forward or backward one week to compare the same day of the week (e.g., comparing one Friday to previous Fridays). The double arrows are only available in day mode.

The bottom panel of the Usage Report page displays the social benefits of energy efficiency; this is discussed below, in the "Motivating Energy Savings" section.

Time Spent Idle

The *Time Spent Idle* page presents the percentage of time spent idle when the computer is on: that is, how much idle time is exhibited compared to active time (see Figure 4). It compares each day in the current week to the same day in the previous week. This report isolates the energy-wasting state the user should be trying to minimize, and identifies whether more idle time is accrued (and more energy wasted) on certain days of the week, or in certain weeks. Navigation buttons at the top allow users to move back and forth to compare earlier weeks. As the chart shows a percentage, it must have data for the full day, so the most recent comparison is for the previous day.

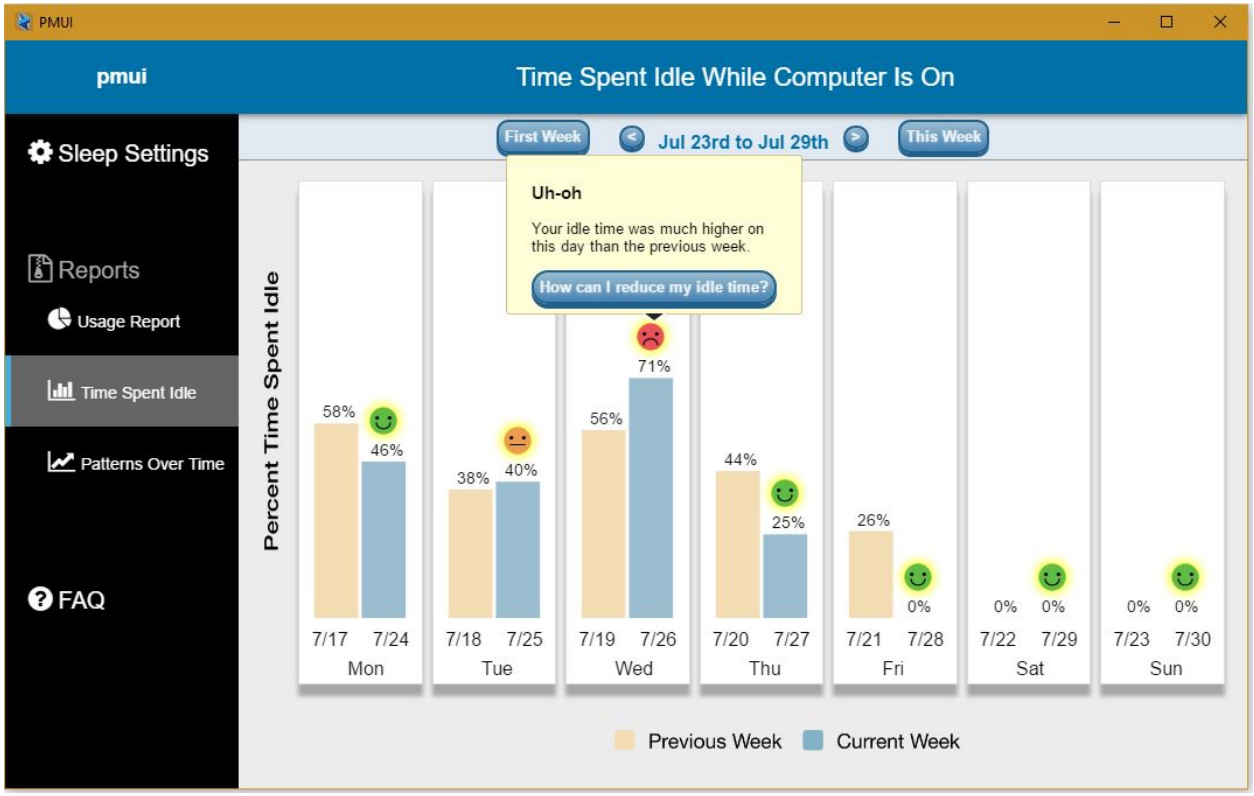
Figure 4: Time Spent Idle Page



Credit: Authors

Depending on the comparison to the idle time on that day in the previous week, the page shows one of three emoticons (big smile, neutral, or big frown). Moving the mouse over the emoticon results in a hover box with a value response and a link to the Sleep Settings page, as shown in Figure 5.

Figure 5: Time Spent Idle Page Displaying Hover Text and Link

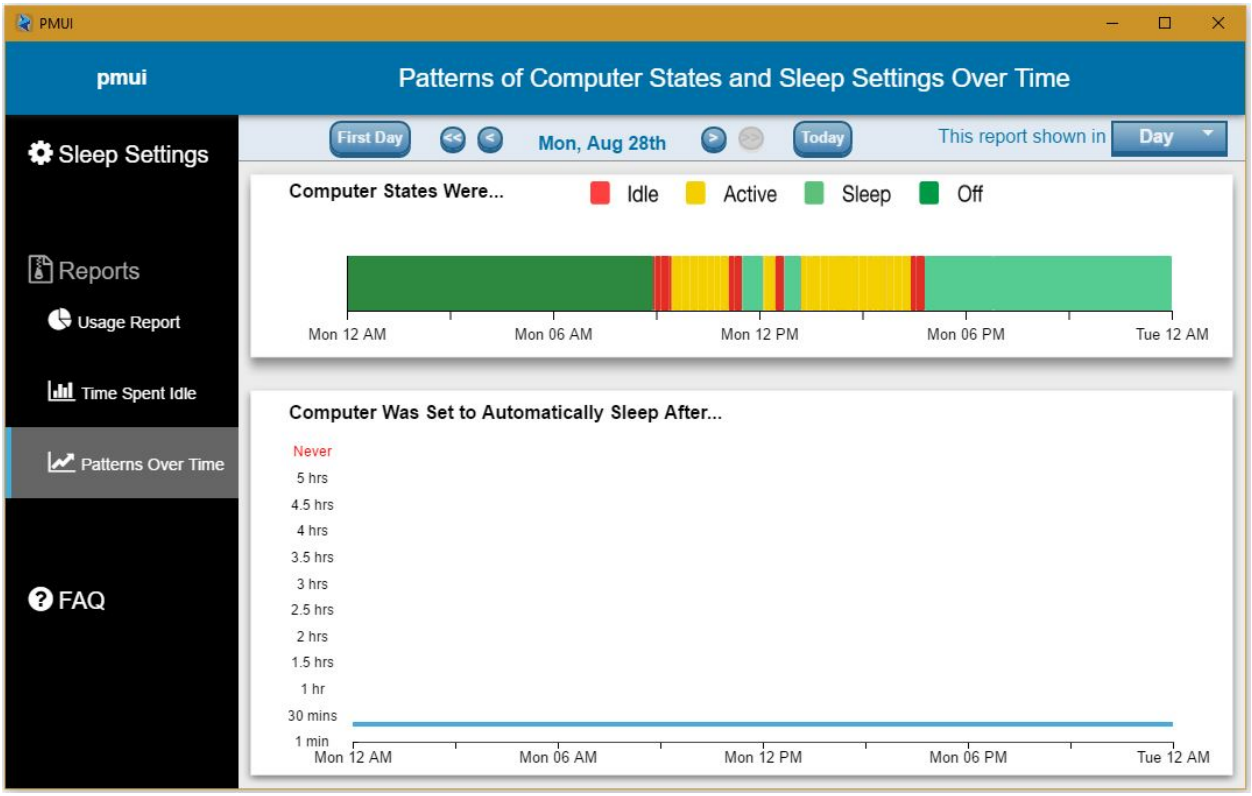


Credit: Authors

Patterns Over Time

The *Patterns Over Time* page has two panels (see Figure 6). The computer states bar is an interval graph showing the periods the computer spent in the four possible states (idle, active, sleep, off) across the time span shown. The sleep settings line graph shows what delay time was set for the computer sleep settings (including “never”) at any given time across the same time span. Showing these two graphs together allows users to compare how a prior change in sleep settings affected the pattern of computer states, for instance, periods spent in idle versus sleep.

Figure 6: Patterns Over Time Page

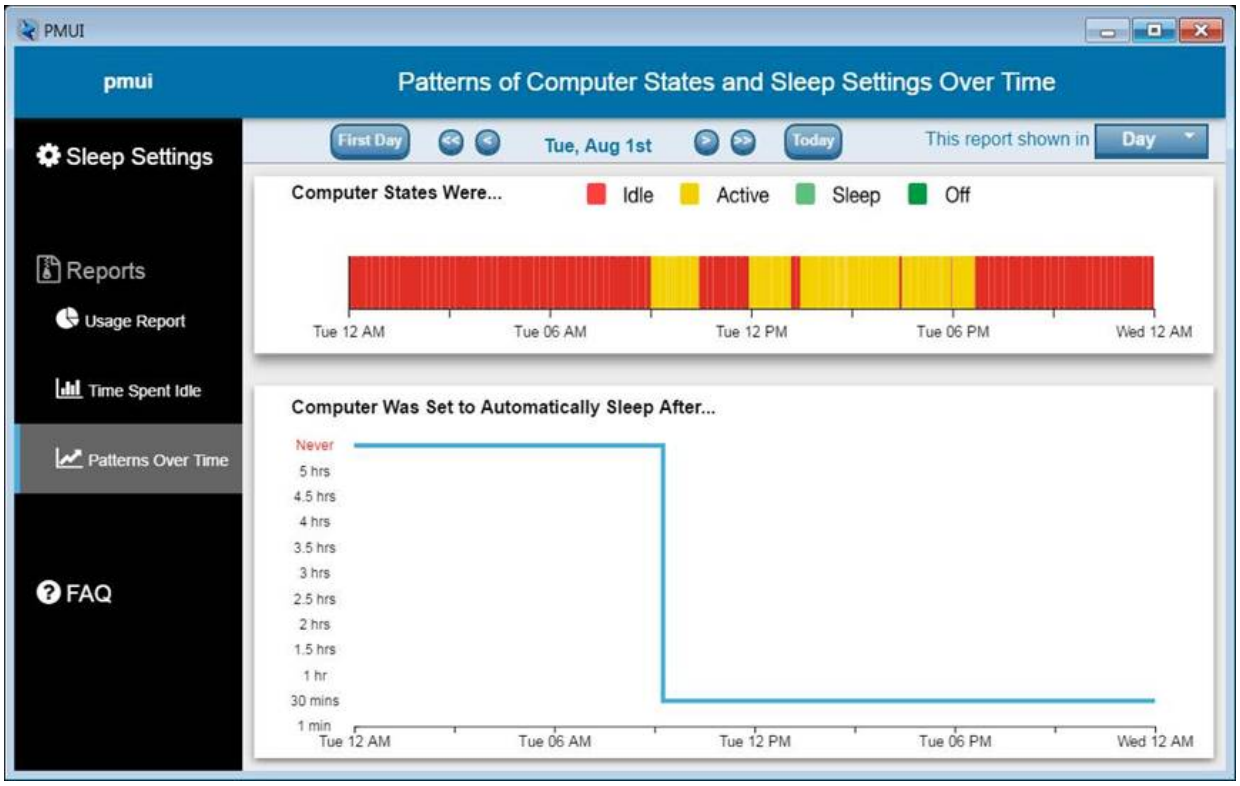


Credit: Authors

This graph can be seen at five levels of granularity: month, week, day, half-day, or hour. Clicking at any point on the states bar zooms the view to the next lowest level (e.g., from day to half-day) to get a closer look at specific transitions between states. As with the other report graphs, navigation buttons at the top can be used to compare across time periods.

The computer states bar also helps users identify periods when their computer is not behaving as expected; for instance, if the computer is waking up in the middle of the night for updates and then not returning to sleep mode, or if it never goes to sleep despite having computer sleep settings enabled (see Figure 7). The current version of the PMUI program does not suggest any solutions for these problems, but providing users with information can spur them to investigate further, or to ask an IT manager for help, as witnessed during the field test.

Figure 7: Example of Identifying Sleep Problems using Patterns Over Time Page

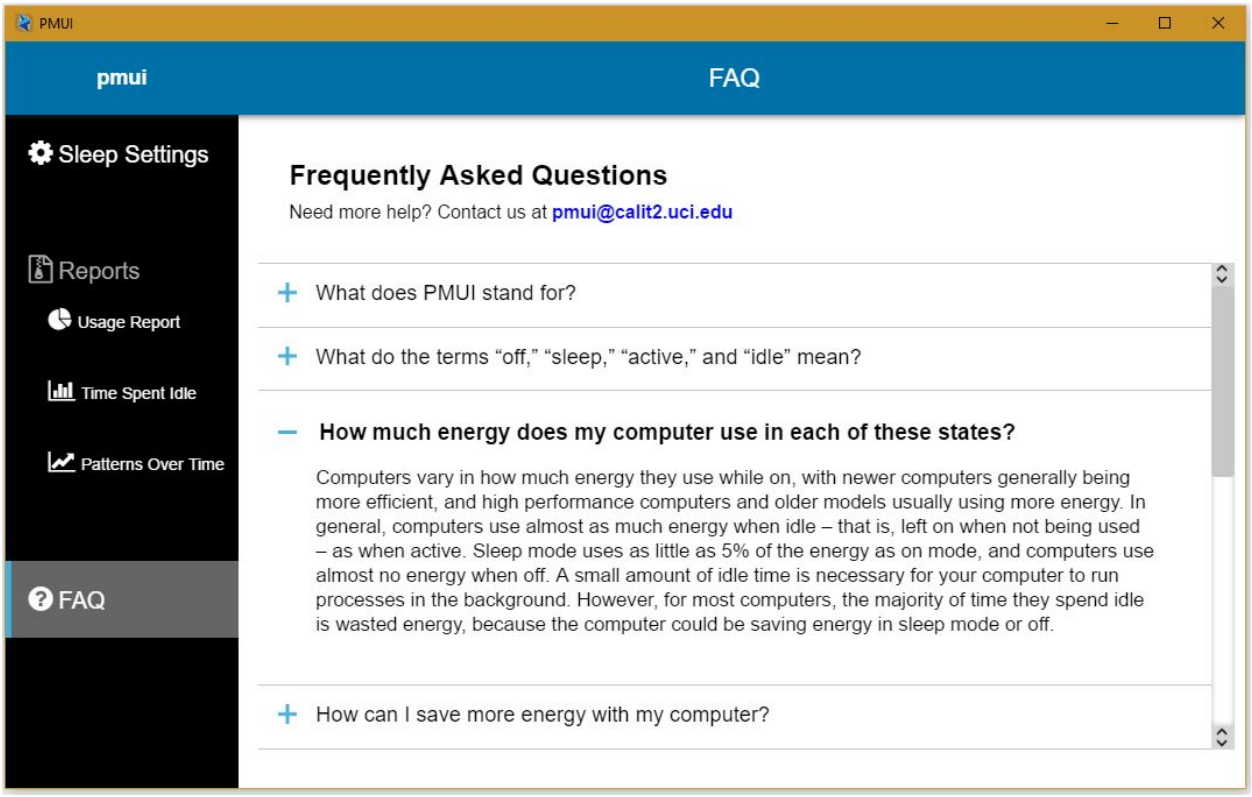


Credit: Authors

Frequently Asked Questions

The last page is a help page titled *Frequently Asked Que*stions, or FAQ (see Figure 8). This section answers questions about saving energy, provides basic documentation to users to clarify how to use the program, and offers an email for submitting questions or complaints. The full list of FAQ questions and answers is included in Appendix A.

Figure 8: Frequently Asked Questions Page



Credit: Authors

Motivating Energy Savings

Communicating Energy Usage

As noted above, the program is limited to presenting information it can collect from the user's computer. As such, it cannot measure and report the computer's actual energy consumption. Instead, the PMUI program focuses on reducing computer idle time, as the computer could potentially be saving energy during these periods by being in a low-power mode. The message that idle time is bad and should be reduced is persistent throughout the user interface. For instance, idle time is represented as red (negative associations), and the Sleep Settings page has a button titled, "How can I reduce my computer's idle time?"

The Frequently Asked Questions (FAQ) page deals more directly with energy savings, including answers to questions such as: "How much energy does my computer use in each of these states?" and "How can I save more energy with my computer?"

Emoticons

Two of the report pages use emoticons (happy or sad faces) to readily convey to users whether their computer is spending too much time idle and to encourage more energy-efficient behavior. On the Usage Report page, the emoticon responds to the comparison between the percentage of idle time observed for the time period shown (e.g., the past week, or a particular day) with the percentage of idle time for the matched Energy Saver Target Profile. There are five

levels: a big smile (up to 3 percentage points higher than the target), a small smile (up to 10 percentage points higher), a neutral or flat smile (up to 20 percentage points higher), a small frown (up to 30 percentage points higher), and a big frown (more than 30 percentage points higher than the target). To further communicate the valence of the icon, the colors of a stoplight are used, consistent with the use of these colors for the pie chart: the happy faces are green, the sad faces are red, and the neutral face is yellow. In the laboratory pretest, subjects did not always notice the emoticon icon on the page, so the programmers added a halo, or highlighting, effect to make it more conspicuous.

On the Time Spent Idle page, the percentage of idle time for each day is compared to the same day in the prior week. Three levels of emoticons are used, depending on whether idle time is lower or higher: a big smile (same or lower), a neutral or flat smile (1 to 5 percentage points higher), and a big frown (more than 5 percentage points higher). The calculation rounded the difference to a whole number, so there are no values between "same" and 1 percentage point higher.

For both reports, the emoticons have hover boxes that clarify the message. For instance, on the Usage Report page, moving the cursor over the emoticon comparing the user's states with the Energy Saver Target Profile produces the following messages, each followed by a link button labeled "Open Sleep Settings":

- Big green smile or small green smile: "Good job! You're a real Energy Saver! Your idle time is similar to the Energy Saver target."
- Neutral yellow face: "Not bad! Your idle time was still higher than the Energy Saver target, though. You could save more energy by reducing your idle time."
- Small red frown or big red frown: "Uh-oh! Your idle time was higher than the Energy Saver target. You could save energy by reducing your idle time."

On the Time Spent Idle page, moving the cursor over the emoticon comparing the idle time for one day compared to the same day last week produces the following messages:

- Big green smile: "Good job! Your idle time was the same or lower on this day than the previous week."
- Neutral yellow face: "Not bad! Your idle time was a little higher on this day than the previous week."
- Big red frown: "Uh-oh! Your idle time was much higher on this day than the previous week."

The neutral and frown faces are followed by a link button labeled "How can I reduce my idle time?" which takes the user to the Sleep Settings page.

Broader Social Benefit Messages

Research indicates that people may be motivated to save energy if they believe their actions have a broader environmental or social benefit (Spence et al. 2014, Asensio and Delmas 2016).

PMUI mentions three such effects: tons of fewer carbon dioxide emissions, the equivalent of how many of acres of trees planted, and the equivalent of how many cars off the road. On the Usage Behavior report, these three effects are shown with easy-to-understand icons (as well as text), with the message: “Practicing positive energy usage would provide the following benefits.” A question mark tooltip explains what the numbers mean in more detail: “The perks: Reducing your current idle time to the Energy Saver Target Profile would save energy. If only 10% of Californians did the same thing, the energy saved per year would be the equivalent of the benefits listed here. Imagine if even more did! Every computer counts!” This message was designed to invoke both a social responsibility norm and a social comparison norm.

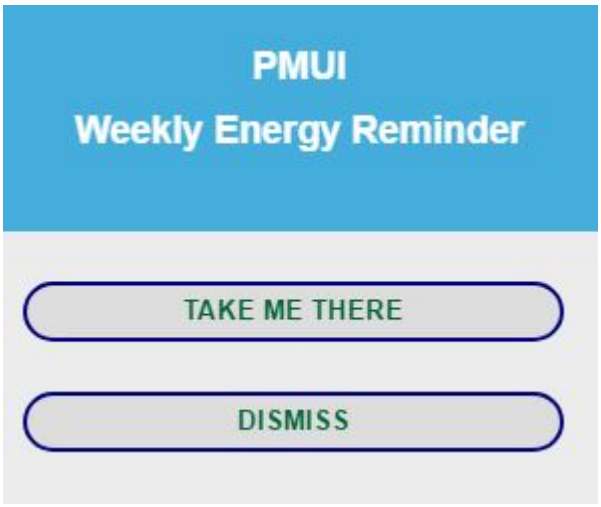
The numbers for the environmental equivalents were based on the ENERGY STAR Computer Power Management Savings Calculator (ENERGY STAR 2018c), which links kWh to the three broader social benefits listed above. ENERGY STAR estimates of power consumption in each mode (on/idle, sleep, and off) for “typical” desktops (not ENERGY STAR-compliant) were used to calculate the estimated kWh for every possible Energy Saver Target Profile, and also for the user’s computer states. Using an estimate of 2,900,000 desktop computers in California, ten percent is 29,000, which is used to multiply the (admittedly small) number of kWh saved per year for each computer to illustrate the broader consequences.

Access and Engagement

Users can access the software at any time using the tray icon or through their program menu; these actions open the program to the sleep settings page. This design was intended to simplify the procedure of reaching the sleep settings, compared to multiple clicks needed for the standard settings in Windows operating systems.

Feedback systems designed to change behavior face the challenge of keeping users engaged with the interface over time without interfering or annoying them. A weekly reminder pop-up was used to encourage continued engagement with the program over time (see Figure 9). The pop-up appears Monday mornings at 10:00 am. If the user clicks on the “take me there” button, PMUI opens to the Usage Report page, set to the previous week (from the previous Monday to yesterday, Sunday). If the user clicks the “dismiss” button, the pop-up disappears. The original plan was to have the pop-up automatically disappear after a certain period of time, but this would mean that users not at their computers at 10:00 would routinely miss the message. An option for users to turn off the reminder function is included, via a check box in the FAQ item titled, “Can I disable the weekly energy reminder popups?”

Figure 9: Weekly Energy Reminder Pop-up



Credit: Authors

Energy Consumption

To supplement data collected by the software during the field test, researchers used study-specific power strips and plug meters to capture energy consumption for each desktop computer and associated monitor(s). This data is not used by the PMUI app, but will be included in the research study as comparison of energy consumption and savings for different computers and setups.

CHAPTER 4:
Software Development

The development of the software was constrained by priorities of the research process and the planned final application, and by standard security protocols of the university.

1. PMUI was designed to look and function similarly on both Windows and Mac operating systems, and to include as many older versions as possible. The team programmed PMUI to function on Windows versions 7, 8, and 10, and on Mac versions from OS X Yosemite to macOS Sierra.
2. PMUI used only free or open source programming components, so that the final version can be more freely distributed and adapted.
3. The research version of PMUI was programmed to transmit data to CalPlug's secure servers, but this function was designed to be easily removed for the future, deployable version.
4. The research version of PMUI was programmed to not automatically display the user interface, but to allow the software to collect data in the background during the baseline data collection period, and to continue to do so for the control subjects.
5. The PMUI application was pretested in a human subjects lab, and changes were made to the interface based on subjects' responses to walk-throughs and comprehension tests.
6. The PMUI team submitted extensive testing results and protocol reports to UCI's Office of Information Technology, who vetted the software before allowing it to be used on university computers.

Data Structure

PMUI requires an intensive data collection framework. The application needs to not only inform users how well they are doing in terms of saving energy, but to report data to the PMUI server that will improve the understanding of the application's performance and the users' habits. The PMUI application collects the following data:

1. Timestamps for when the computer transitions to each state: idle, active, sleep, and off.
2. Any changes in power management settings (computer sleep and display sleep).
3. Current delay set for computer and display sleep settings (in minutes).
4. When the user runs the app, and when it's closed.
5. When specific pages of the PMUI app are accessed by the user.
6. When the user toggles the "Temporarily Disable Sleep Settings" box on the Sleep Settings page.
7. How long the sleep settings were temporarily disabled using the "Temporarily Disable Sleep Settings" feature.
8. Internal processes of the PMUI software, such as when the user interface was activated and whether and when the weekly reminder pop-up was sent.
9. Processing computer state data so that the charts can be generated.

In determining computer states, no distinction is made between short-idle or long-idle. There are two levels of idle measured. For the purposes of research, idle is measured as any time the computer is on but there has been no keystroke or mouse activity for one minute. Thus, it

captures short-idle as well as long-idle, although they are grouped together in the data. For the purposes of calculating and displaying results to the user for the Usage Report and Time Spent Idle pages, idle is measured as no keystroke or mouse activity for ten minutes. This decision was made to provide a cushion so that users did not feel overly penalized for short-idle time that naturally occurs during the course of active use. For the Patterns Over Time page, periods are presented down to the minute.

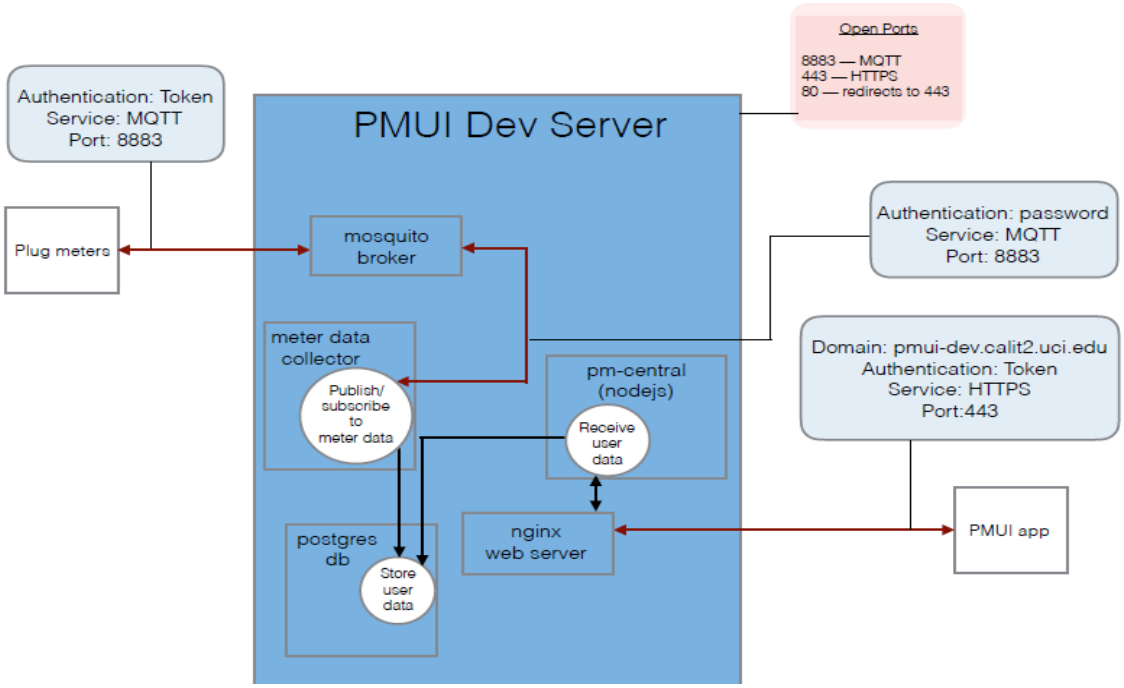
The software also does not distinguish between the various levels of sleep outlined by the standards of the Advanced Configuration and Power Interface (ACPI), counting the initial transition into sleep mode only. Unfortunately, hibernation (ACPI state S4) could not be distinguished from sleep (ACPI state S3).

PMUI was programmed to collect data on what programs might be preventing the computer from going to sleep if sleep is enabled. In some cases, this may indicate user activity, such as when users are watching a video. In other cases, it may indicate a bug, or some other sleep blocking activity. For Windows OS computers, there is a command line query to ask the Windows OS specifically for what processes are blocking sleep. However, the query requires administrator privileges every time it is run, and thus could not be implemented in the software. Instead, if the computer sleep settings are enabled and yet the computer has been idle longer than the set delay time, the PMUI program records which programs are currently running in Task Manager. For Mac OS computers, PMUI is able to record which program(s) are preventing the computer from transitioning to sleep. This feature collected data about "sleep block" events during the field test. However, subsequent data analyses identified many periods of long idle when computers should have been in sleep mode, suggesting a problem with this feature that the team has not yet been able to address.

Data Flow

The data collected by the software applications and the plug meters is managed by the PMUI central server. The server uses several services and technologies, such as PostgreSQL, MySQL, MQTT, NodeJS, JavaScript and Python, to collect and store the information needed (see Figure 10). All data transfer are encrypted through Secure Sockets Layer (SSL).

Figure 10: Data Flow



Credit: Authors

Desktop Application

User Interface

The design of the user interface was chosen based on current research on user behavior intervention studies and best practices in the field of human-computer interaction. The main features included in the design of the applications are as follows:

1. The interface is easy to access (while remaining non-intrusive) and easy to use (including focusing only on computer sleep and display sleep);
2. It encourages positive intrinsic motivation for energy efficient behaviors;
3. It provides direct and timely feedback to the user on how their selected settings affect their computer's idle time and thus, energy use.

As described earlier, the interface consists of five pages: a page for sleep settings, three pages that report the user's data, and a FAQ page. On the report pages, the data collected by the PMUI app is used to generate charts and graphs as follows:

1. Pie Chart: used to show the percentages of idle, active, sleep, and off time in a given period.
2. Bar Graph: used to display time spent idle on each day of the week compared to the previous week.
3. Interval Graph: used to display the pattern of computer states over a given period.
4. Line graph: used to show the sleep delay time setting over a given period.

Software Implementation

The application was developed based on the design of the graphical user interface and the requested features. The application links to the native Windows or Mac operating system's power management in a bi-directional way, so that a settings change made using either interface (native or PMUI) is reflected in the other.

There are two major parts to the PMUI application: the graphical user interface (GUI) and the event listener subprocess. The PMUI GUI was built using Electron, a cross-platform desktop application framework. The main technologies used were HTML, CSS (frontend application), and JavaScript. The PMUI GUI application plays many roles: it administers and communicates with the event listener subprocess, it sends data to the server, and it displays an interactive interface to the user. The interactive interface has five pages: Settings, Usage Report, Time Spent Idle, Patterns Over Time, and FAQ. All of the slider and chart components are built from the D3 (Data-Driven Documents) JavaScript API. The event listener subprocess is a console application, written in C# for Windows and Swift for Mac, that runs in the background to collect computer state data. The event listener for Windows uses Window's powercfg terminal commands to get and set the user's display and computer sleep settings and determine what programs are blocking sleep. The event listener for Mac uses Mac's pmset terminal command to get and set the user's display and computer sleep settings and determine what programs are blocking sleep. Because pmset requires administrator privileges, the PMUI program obtains these permissions during install.

NodeJS and Python libraries were also used for the backend application. For an exhaustive list of libraries used by the PMUI app, see Appendix B.

The PMUI application uses a few of the standard JavaScript community tools for debugging and testing and for code quality. The current repositories of the applications are stored in a UCI OIT GitHub at: https://github.oit.uci.edu/ETAD/pm-shell that contains the user side of the PMUI application, and https://github.oit.uci.edu/ETAD/pmcentral, which contains the server side of the PMUI framework.

Software Issues

Missing data when logged out

Testing the PMUI software on a wide range of computers and operating systems produced only brief periods of time (usually less than one minute) when the computer state was unknown, such as when the computer had just been turned on. However, none of the tests were performed on computers where users logged in with a user profile. The software must be installed on the user's profile in order to access the power management settings. However, when users logged off for the evening, PMUI was unable to access the computer to collect state data. This issue went unnoticed until the first round of treatment subjects had their user interface activated and later reported seeing long periods of gray (unknown) in their reports. The software team updated the software to read the system events from the OS system logs retroactively every time PMUI is launched, to fill in the missing sleep, off, and idle events.

Weekly reminder pop-ups

The reminders were intended to appear every Monday morning, and encourage subjects to click on the button to open the Energy Report page. They could instead click on a button to dismiss the message. Some of the first wave of subjects, faced with questions in the RV3 survey about various features of the software, reported that they had never seen the weekly reminder pop-up. Other subjects reported that the pop-up appeared for only a few seconds and then disappeared before they could notice it and click on it. This was determined to be an artifact of using the native operating system notifications. An updated version of the software used a custom built notification to overcome this issue.

False idle periods

Some subjects reported observing idle periods in the Pattern Over Time page when they should have seen sleep periods. These proved difficult to diagnose, because some supposedly false idle periods were indeed idle periods, caused by computers that did not transition to sleep when they were supposed to. However, other false idle periods should have been coded as sleep; these were traced to sleep and idle events with the same timestamp, leading to PMUI miscoding the result. An updated version of the software addressed this by ignoring a false idle event if it occurred at the same time as a sleep event.

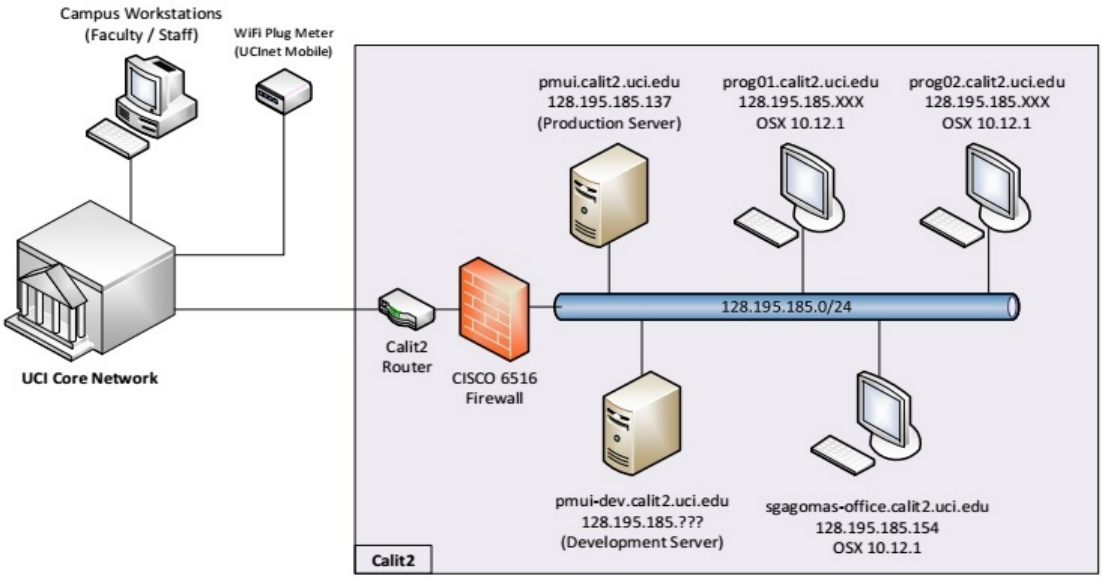
Sleep block tracker issues

As mentioned above, the sleep block tracker feature appears to have accurately recorded some events when computers stayed idle when they should have transitioned to sleep, but missed many others. This feature was included for data collection purposes and was not part of the user interface, so it had no effect on the user experience during the field test. As the problem was not discovered until the data analysis phase, it was not fixed in the updates and has yet to be addressed.

PMUI Network

The data network for PMUI that was used in the current study is shown in Figure 11. The data collected from the PMUI app and plug meters was transmitted to the PMUI central server in a secure and encrypted network. All data transmission occurred within the UCI Network. The PMUI Server was located in the Calit2 building, in one of the server rooms. VMWare virtualization was used to run two servers, one for development and one for production. Images were stored in a secure network server. The PMUI servers were Linux-based.

Figure 11: PMUI Network



Credit: Authors

CHAPTER 5:
Technology Transfer Activities

CalPlug and Calit2 Outreach

CalPlug has been hosting annual spring workshops at UCI's California Institute for Telecommunications and Information Technology (Calit2). The CalPlug Workshop draws a small group of researchers, academics, policy makers, advocates, and representatives from industry, many of whom have ongoing relationships with CalPlug. The PMUI project was presented at the CalPlug Workshop in April 2016 (introduction of the project and prototype presentation), April 2018 (report of preliminary sleep settings results), and April 2019 (sleep settings, changes to computer states, and energy savings).

Calit2 and CalPlug regularly receive visitors (academic researchers, scientists, and administrators, and representatives from industry and utilities) and give tours. Since fall of 2017, these tours have included a demo of the PMUI app, along with a summary of the field test and its results.

Calit2's award-winning *Interface Magazine* is scheduled to publish an article summarizing the project, its main results, and advice to computer users, including appropriate graphics and illustrations, as the cover feature for the Fall 2019 issue. The external audience includes industry, entrepreneurs, investors, other universities, government agencies and non-profit organizations; internal audience includes senior UCI administrators and academic leadership, campus communicators and targeted faculty. This article will also be distributed to the subjects of the study. The Interface Magazine is published twice a year; the Fall 2019 issue is the next issue.

An article summarizing the project, its main results and advice to computer users will be posted on both the Calit2 and CalPlug websites. The website is visited frequently by industry partners, faculty and students.

A media advisory will be written that summarizes the project and noteworthy results and directs interested readers to the website article and PMUI webpage. The release will target specialized news organizations who cover energy efficiency, sustainability, consumer behavior and tech sectors.

Outreach to the Scientific and Professional Community

The PMUI project has produced far more results than can be presented in a single presentation, paper, or journal article. The basic findings must be repeated in each paper or presentation, but each one will focus more thoroughly on a different aspect of the data, to provide a unique set of findings and interpretations for each venue. So far, one paper featuring the preliminary results of 288 subjects has been presented and then published in the conference proceedings of the American Council for an Energy-Efficient Economy (ACEEE) 2018 conference (Pixley, Gago Masague, and Fallman 2018). ACEEE's Summer Study on Energy Efficiency in Buildings is a

biennial conference that attracts a diverse group of academics and other professionals to discuss cutting-edge technologies, strategies, and programs for reducing energy use and addressing climate impacts. At the Behavior, Energy & Climate Change (BECC) conference, the PMUI app was presented with power management behavior results from the full sample (Pixley et al. 2018). BECC is an annual conference focused on user behavior and decision making as it relates to energy-efficient and low-carbon interventions. Attendees include energy efficiency practitioners, researchers, and utilities. A paper that summarizes the main findings of user power management behavior has been submitted to a leading journal and is currently under review. There are more papers planned, with a focus on energy savings, user experiences, usage patterns and persistence, and barriers to power management.

Outreach to Distribute Technology

The PMUI app was developed using open source (free) software, to better facilitate distribution to a wide range of users. It has been modified to be used for non-research purposes, but would benefit from additional modifications to address issues discovered during the field test. As with any software, distribution of PMUI to individuals requires an agent(s) committed to maintaining the software and monitoring it for standard security updates.

The beta version of the standalone PMUI application is available for free on its own page on the Calit2 website (pmui.calit2.uci.edu). The availability of the app will be communicated through Calit2's extensive contact network and beyond, through the planned press release and *Interface* article. CalPlug is also in discussions with the Office of Information Technology and the Office of Sustainability about the possibility of distributing the software more broadly across campus, whether in research mode or standalone mode.

For the near-term, CalPlug will maintain and monitor the software for security and functionality and will seek independent resources to further improve the app based on feedback and problems discovered during the field test, and update the publicly available version.

CalPlug will also make the software available to other groups who want to use it as-is or to further develop it on their own. CalPlug has been demonstrating the software and its capabilities to multiple audiences, including visitors to CalPlug and the annual CalPlug Workshop and by presenting research results at conferences. Many expressed provisional interest in the program. CalPlug will continue to reach out to multiple potential agents, including government, university, utility, and commercial enterprises. The software and all relevant code will be made available to any such enterprise who wishes to adopt the technology; they may then modify the software to customize it for their own usage.

For the longer term, CalPlug plans to reach out to Microsoft, Apple, and computer manufacturers, with the goal of incorporating PMUI prior to distribution. It is assumed that this will be more effective once the software has been distributed more widely and broad interest has been established. It should be noted that even if one or both companies adopt the app, the baseline software remains open source and can continue to be used, updated, and customized by others.

Other approaches to contacting and engaging potential partners may also be explored.

CHAPTER 6: Conclusions

This report summarizes the design and development of the Power Management User Interface (PMUI) software application. PMUI functions on a range of Windows and Mac operating systems. After extensive testing before the field test, as well as continual improvements during the field test, the software is fully functional for the purposes of a research platform. However, some changes are necessary prior to distribution and further use:

- As with any software application, before further testing or distribution, PMUI must be updated to reflect any external changes since the last version, such as updates to Windows or Mac protocols.
- The data collection and report generation features functioned well during the field test. A few researchers retained the software on their own computers, providing a test period substantially longer than the three to four months of most subjects. The field test version of the PMUI software was only designed to accumulate and process up to six months of data. Although the program is still operating on several computers after more than a year, it exhibits long delays on opening and occasional grayed-out reports. A public version of PMUI would require implementing optimized data storage and retrieval methods to accommodate long-term use.
- In order to analyze the effectiveness of the user interface during the planned field test study, the current version of the software was designed to transmit data on computer states and user activity to a secure server managed by the research team. For any non-research deployment, PMUI must first be modified to remove this research mode.
- One bug was identified toward the end of the field test which the software development team has not been able to solve. When computers have very short delay times before transitioning to sleep (less than five minutes), PMUI shuts down before registering the sleep event, and subsequent time is recorded as idle. This does not affect many computers, but it should be addressed.

Other modifications or additions are optional but recommended by the research team. Field test subjects were asked for feedback about the various functions and pages of the user interface, and the program collected data about how often the application's functions and pages were used. The research team also made its own observations of the functionality of the interface, and received valuable suggestions from the Technical Advisory Committee that could not be incorporated into the current version in the current scope. Based on this information, CalPlug recommends the following revisions of the user interface that could more fully engage and encourage users:

- Program the frequency of the weekly reminders to be dynamic, responding to whether the user has recently looked at the PMUI program and whether the sleep settings are enabled; this should reduce the chance that users are annoyed by frequent reminders even when their sleep settings are already efficient. Consider also flexible timing, as some users may prefer reminders on other days or the week.

- Rethink the Time Spent Idle page. Users may not understand presenting the percentage of time idle as a function of the amount of time on. Also, as usage and idle time vary day by day and the comparison criteria is narrow, this page often presents neutral or sad emoticons even when overall usage is efficient.
- Consider the Technical Advisory Committee's suggestion of adding predictive feedback on settings: for example, predictions of how idle time would change depending on various sleep setting delay times the user is considering.
- Rethink how the environmental impact rewards on the Usage Report page are communicated. In the current version, the numbers represent the benefits users could make if they reduced their idle time, but this means the benefits appear greatest to the most inefficient users and disappear when users are already efficient. An alternate approach could reward users if they are meeting the Energy Saver Target profile goal by presenting higher environmental benefits.
- Give users actionable information on troubleshooting and solving problems with computer sleep. The Patterns over Time page can identify problem periods when the computer is not sleeping even though sleep settings are enabled. It may be possible for PMUI to tell the user which background process is keeping the computer from transitioning to sleep.

The current version is designed for desktop computers, which were the focus of the field test. The same interface could be adapted for laptop computers with minor modifications to the sleep settings page, to distinguish settings while plugged in versus while on battery power. However, user behavior for laptop computers is expected to differ enough from that of desktop computers that additional field testing would be recommended to estimate energy savings.

The intended end product of the required changes would be a stand-alone application that can be distributed freely and used in a wide range of residential and enterprise settings. As with any application, it would require a dedicated team to maintain the integrity of the software, adapting it against new security threats and keeping it compatible with operating system and other upgrades.

CHAPTER 7:
Benefits to Ratepayers

The PMUI software was designed to be used on any personal desktop, including residential and business. It was developed using free or open source software to facilitate its use and distribution. The goal is to distribute the software freely to the public, after taking the required steps to stabilize the software listed above (and perhaps also the recommended steps to improve the software).

The PMUI software was designed to be used on any personal desktop, including residential and commercial, expanding its potential reach to California ratepayers. The PMUI software can benefit California ratepayers both directly and indirectly. Ratepayers who use the software will learn more about power management and how to efficiently use their computers, and depending on their current settings, could save energy and money. All ratepayers will indirectly benefit from the energy that is saved by others who use the program, through reduced carbon dioxide emissions and reduced demand on the energy grid. Estimates of the effect of using PMUI are detailed in the Final Project Report, which presents the results of the field test (Pixley et al. 2019). At a projected full deployment of ten percent of California desktops, PMUI would save 16,245 MWh/year and over $2.6 million in electricity costs, and would reduce carbon dioxide emissions by 11,488 metric tons.

This research sets the groundwork for additional projects, applying the same behavioral feedback intervention strategies to laptops and to other plug load devices in homes and offices.

GLOSSARY

Term/Acronym	Definition
ACPI	Advanced Configuration and Power Interface, an industry-wide open standard for managing computer devices and components, including power management
CA	California
CalPlug	The California Plug Load Research Center, at the University of California, Irvine
CEC	California Energy Commission
Computer, desktop	Unless otherwise noted, the term "computer" refers to desktop computers, including all-in-one models, and including both Windows and Mac computers. The term "desktop" is used in preference to the term "PC" to avoid implications about the operating system in use.
EPIC (Electric Program Investment Charge)	The Electric Program Investment Charge, created by the California Public Utilities Commission in December 2011, supports investments in clean energy technologies that benefit electricity ratepayers of Pacific Gas and Electric Company, Southern California Edison Company, and San Diego Gas & Electric Company.
kWh	Kilowatt hour, a standard unit of energy equivalent to one kilowatt of power sustained for one hour. The kWh is commonly used as a billing unit for energy by electric utilities.
OIT	Office of Information Technology, a unit at UCI
OS	Operating system
Power Management, PM	Power management refers to a range of features that transition the computer, specific components, or an attached monitor into low-power modes. This includes basic automatic settings that transition the computer or monitor into sleep mode after a set delay period and advanced automatic settings such as those that transition the computer to hibernate or turn off the hard disk after a set delay period. It also includes manual power management options, in which the user directly transitions the computer to sleep, hibernate, shutdown, or some other low-power mode through menu or keyboard commands.
PMUI	Power Management User Interface, the working title for the software developed in the current project.
UCI or UC Irvine	University of California, Irvine

User	The subject who uses the computer in question (who may or may not be the owner).
User interface	The part of a software application that the user can see and interact with.
TWh	Terawatt hour, a measure used for metering large amounts of electrical energy, such as when discussing annual energy consumption. 1 TWh is 10^{12} * 1 Wh (watt hour), or 10^{4} * 1 kWh (kilowatt hour).

REFERENCES

1E, and Alliance to Save Energy. 2009. PC Energy Report 2009: United States, United Kingdom, Germany.

Abrahamse, Wokje, Linda Steg, Charles Vlek, and Talib Rothengatter. 2005. "A review of intervention studies aimed at household energy conservation." *Journal of Environmental Psychology* 25 (3):273-291. doi: http://dx.doi.org/10.1016/j.jenvp.2005.08.002.

Acker, Brad, Carlos Duarte, and Kevin Van Den Wymelenberg. 2012. "Office Plug Load Savings: Profiles and Energy-Saving Interventions." ACEEE Summer Study on Energy Efficiency in Buildings, Asilomar, California, August 12-17.

Armel, K.Carrie, Abhay Gupta, Gireesh Shrimali, and Adrian Albert. 2013. "Is disaggregation the holy grail of energy efficiency? The case of electricity." *Energy Policy* 52:213-234. doi: http://dx.doi.org/10.1016/j.enpol.2012.08.062.

Asensio, O. I., and M. A. Delmas. 2015. "Nonprice incentives and energy conservation." *Proceedings of the National Academy of Sciences* 112 (6):E510-5. doi: 10.1073/pnas.1401880112.

Asensio, Omar Isaac, and Magali A. Delmas. 2016. "The dynamics of behavior change: Evidence from energy conservation." *Journal of Economic Behavior & Organization* 126, Part A:196-212. doi: https://doi.org/10.1016/j.jebo.2016.03.012.

Barr, Michael, Chris Harty, and Jane Nero. 2010. Thin Client Investigation Including PC and Imaging State Data. Pacific Gas and Electric Company.

Bensch, Ingo, Scott Pigg, Karen Koski, and Rana Belshe. 2010. Electricity Savings Opportunities for Home Electronics and Other Plug-In Devices in Minnesota Homes: A Technical and Behavioral Field Assessment. Madison, Wisconsin: Energy Center of Wisconsin.

Bird, Stephen, and Lisa Legault. 2018. "Feedback and Behavioral Intervention in Residential Energy and Resource Use: a Review." *Current Sustainable/Renewable Energy Reports* 5 (1):116-126.

Blasch, Julia, Massimo Filippini, and Nilkanth Kumar. 2017. "Boundedly rational consumers, energy and investment literacy, and the display of information on household appliances." *Resource and Energy Economics*. doi: https://doi.org/10.1016/j.reseneeco.2017.06.001.

Boston University. 2018. "Ten Sustainable Actions." accessed Accessed October 15, 2018. http://www.bu.edu/sustainability/what-you-can-do/ten-sustainable-actions.

Buchanan, Kathryn, Riccardo Russo, and Ben Anderson. 2015. "The question of energy reduction: The problem(s) with feedback." *Energy Policy* 77:89-96. doi: https://doi.org/10.1016/j.enpol.2014.12.008.

Byrne, David P., Andrea La Nauze, and Leslie A. Martin. 2018. "Tell Me Something I Don't Already Know: Informedness and the Impact of Information Programs." *Review of Economics and Statistics*. doi: https://doi.org/10.1162/REST_a_00695.

Canfield, Casey, Wändi Bruine de Bruin, and Gabrielle Wong-Parodi. 2017. "Perceptions of electricity-use communications: Effects of information, format, and individual differences." *Journal of Risk Research* 20 (9):1132-1153. doi: http://dx.doi.org/10.1080/13669877.2015.1121909.

Chetty, Marshini, A.J. Bernheim Brush, Brian R. Meyers, and Paul Johns. 2009. "It's not easy being green: understanding home computer power management." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Boston, MA, USA.

City of Irvine. 2018. "Office Equipment and Energy Use." accessed November 19, 2018. http://www.cityofirvine.org/environmental-programs/conserve.

Darby, Sarah. 2006. The Effectiveness of Feedback on Energy Consumption. Environmental Change Institute University of Oxford.

Delmas, Magali A., Miriam Fischlein, and Omar I. Asensio. 2013. "Information strategies and energy conservation behavior: A meta-analysis of experimental studies from 1975 to 2012." *Energy Policy* 61:729-739. doi: http://dx.doi.org/10.1016/j.enpol.2013.05.109.

Delmas, Magali A., and William Kaiser. 2014. Behavioral responses to real-time individual energy usage information: A large scale experiment. Prepared for the California Air Resources Board and the California Environmental Protection Agency.

Ehrhardt-Martinez, Karen, Kat A. Donnelly, and John A. Skip Laitner. 2010. Advanced Metering Initiatives and Residential Feedback Programs: A Meta-Review for Houshold Electricity-Saving Opportunities. Washington, D. C.: American Council for an Energy-Efficient Economy.

Electric Power Research Institute. 2009. Residential Electricity Use Feedback: A Research Synthesis and Economic Framework. Palo Alto, CA.

ENERGY STAR. 2013. "Low Carbon IT Campaign Kicks Off 2013 with Organizations Pledging to Power Manage 360,000 Computers." accessed November 19, 2018. https://www.energystar.gov/about/content/low-carbon-it-campaign-kicks-2013-organizations-pledging-power-manage-360000-computers.

ENERGY STAR. 2018a. "6 Ways to reduce IT Energy Costs." accessed November 19, 2018. https://www.energystar.gov/products/reduceitenergycosts

ENERGY STAR. 2018b. "Activate Power Management on Your Computer." accessed November 19, 2018. https://www.energystar.gov/products/low_carbon_it_campaign/power_management_computer.

ENERGY STAR. 2018c. "ENERGY STAR Computer Power Management Savings Calculator." accessed November 19, 2018. https://www.energystar.gov/sites/default/files/asset/document/LowCarbonITSavingsCalc.xlsx.

ENERGY STAR. 2018d. "Pledge to Save Energy." accessed November 19, 2018. https://www.energystar.gov/campaign/takeThePledge.

Ferraro, Paul J., and Michael K. Price. 2013. "Using Nonpecuniary Strategies to Influence Behavior: Evidence from a Large-Scale Field Experiment." *Review of Economics and Statistics* 95 (1):64-73.

Fischer, Corinna. 2008. "Feedback on household electricity consumption: a tool for saving energy?" *Energy Efficiency* 1 (1):79-104. doi: 10.1007/s12053-008-9009-7.

Froehlich, Jon. 2009. "Promoting Energy Efficient Behaviors in the Home through Feedback: The Role of Human-Computer Interaction." HCIC 2009 Winter Workshop.

Gupta, Abhay, and Prateek Chakravarty. 2013. Impact of Energy Disaggregation on Consumer Behavior. Sunnyvale, CA.

Hackel, Scott, Chris Plum, and Rick Carter. 2016. "Advancing the Last Frontier – Reduction of Commercial Plug Loads." 2016 ACEEE Summer Study on Energy Efficiency in Buildings, Pacific Grove, California.

Hermsen, Sander, Jeana Frost, Reint Jan Renes, and Peter Kerkhof. 2016. "Using feedback through digital technology to disrupt and change habitual behavior: A critical review of current literature." *Computers in Human Behavior* 57:61-74. doi: https://doi.org/10.1016/j.chb.2015.12.023.

Intel. 2009. Energy Star* Version 5.0 System Implementation. Published by Intel with technical collaboration from the U.S. Environmental Protection Agency.

Jain, Rishee K., John E. Taylor, and Gabriel Peschiera. 2012. "Assessing eco-feedback interface usage and design to drive energy efficiency in buildings." *Energy and Buildings* 48:8-17. doi: http://dx.doi.org/10.1016/j.enbuild.2011.12.033.

Karjalainen, Sami. 2011. "Consumer preferences for feedback on household electricity consumption." *Energy and Buildings* 43 (2-3):458-467. doi: http://dx.doi.org/10.1016/j.enbuild.2010.10.010.

Karlin, Beth. 2011. "Tracking and learning: exploring dual functions of residential energy feedback." Proceedings of the 6th International Conference on Persuasive Technology:

Persuasive Technology and Design: Enhancing Sustainability and Health, Columbus, Ohio, USA.
Karlin, Beth, Joanne F. Zinger, and Rebecca Ford. 2015. "The effects of feedback on energy conservation: A meta-analysis." *Psychological Bulletin* 141 (6):1205-1227. doi: 10.1037/a0039650.
Kluger, Avraham N., and Angelo DeNisi. 1996. "The effects of feedback interventions on performance: A historical review, a meta-analysis, and a preliminary feedback intervention theory." *Psychological Bulletin* 119 (2):254-284. doi: 10.1037/0033-2909.119.2.254.
Korn, David, Robert Huang, David Beavers, Thomas Bolioli, and Mike Walker. 2004. "Power management of computers - $ 1.5 billion in potential energy savings annually." Proceedings of the 2004 IEEE International Symposium on Electronics and the Environment, May 10, 2004 - May 13, 2004, Scottsdale, AZ, United States.
Krishnamurti, Tamar, Alexander L. Davis, Gabrielle Wong-Parodi, Jack Wang, and Casey Canfield. 2013. "Creating an in-home display: Experimental evidence and guidelines for design." *Applied Energy* 108:448-458. doi: http://dx.doi.org/10.1016/j.apenergy.2013.03.048.
Lesic, Vedran, Wändi Bruine de Bruin, Matthew C. Davis, Tamar Krishnamurti, and Inês M. L. Azevedo. 2018. "Consumers' perceptions of energy use and energy savings: A literature review." *Environmental Research Letters* 13 (3):033004.
Lynham, John, Kohei Nitta, Tatsuyoshi Saijo, and Nori Tarui. 2016. "Why does real-time information reduce energy consumption?" *Energy Economics* 54:173-181. doi: https://doi.org/10.1016/j.eneco.2015.11.007.
McCalley, L. T., and Cees J. H. Midden. 2002. "Energy conservation through product-integrated feedback: The roles of goal-setting and social orientation." *Journal of Economic Psychology* 23 (5):589-603. doi: https://doi.org/10.1016/S0167-4870(02)00119-8.
Mercier, Catherine, and Laura Moorefield. 2011. Commercial Office Plug Load Savings Assessment. Sacramento, CA: California Energy Commission.
Moorefield, Laura, Brooke Frazer, and Paul Bendt. 2011. Office Plug Load Field Monitoring Report. Sacramento, CA: California Energy Commission.
Murtagh, Niamh, Michele Nati, William R. Headley, Birgitta Gatersleben, Alexander Gluhak, Muhammad Ali Imran, and David Uzzell. 2013. "Individual energy use and feedback in an office setting: A field trial." *Energy Policy* 62:717-728. doi: https://doi.org/10.1016/j.enpol.2013.07.090.
Murugesan, Latha Karthigaa, Rashina Hoda, and Zoran Salcic. 2015. "Design criteria for visualization of energy consumption: A systematic literature review." *Sustainable Cities and Society* 18:1-12. doi: https://doi.org/10.1016/j.scs.2015.04.009.
Navigant Consulting. 2013. Analysis and Representation of Miscellaneous Electric Loads in NEMS. Washington, D.C.: U.S. Energy Information Administration.
Orland, B., N. Ram, D. Lang, K. Houser, N. Kling, and M. Coccia. 2014. "Saving energy in an office environment: A serious game intervention." *Energy and Buildings* 74:43-52. doi: 10.1016/j.enbuild.2014.01.036.
Picklum, Roger E., Bruce Nordman, and Barbara Kresch. 1999. Guide to Reducing Energy Use in Office Equipment. San Francisco, CA: Bureau of Energy Conservation.
Pixley, Joy E., Sergio Gago Masague, and Raquel Fallman. 2018. "Field Test of a New User Interface for Computer Sleep Settings." ACEEE Summer Study on Energy Efficiency in Buildings, Pacific Grove, California.
Pixley, Joy E., Sergio Gago Masague, Raquel Fallman, Sabine Kunrath, and Guann-Pyng Li. 2018. "A Feedback App that Improves Use of Computer Sleep Settings: Field Test Results." Behavior, Energy & Climate Change (BECC), Washington, D. C.
Pixley, Joy E., Sabine Kunrath, Sergio Gago Masague, Raquel Fallman, and G.P. Li. 2019. The Power Management User Interface Study. Sacramento, CA: California Energy Commission.
Pixley, Joy E., and Stuart A. Ross. 2014. Monitoring Computer Power Modes Usage in a University Population. Sacramento, CA: California Energy Commission.

Pixley, Joy E., Stuart A. Ross, Ankita Raturi, and Alan C. Downs. 2014. A Survey of Computer Power Modes Usage in a University Population. Sacramento, CA: California Energy Commission.
Pollard, Carol Elaine. 2016. "Up close and personal." *International Journal of Sustainability in Higher Education* 17 (1):68-85. doi: 10.1108/IJSHE-03-2014-0043.
Promann, Marlen, and Sabine Brunswicker. 2017. "Affordances of Eco-Feedback Design in Home Energy Context." Twenty-third Americas Conference on Information Systems (AMCIS), Boston.
Roberson, Judy A., Carrie A. Webber, Marla C. McWhinney, Richard E. Brown, Margaret J. Pinckard, and John F. Busch. 2004. After-hours Power Status of Office Equipment and Inventory of Miscellaneous Plug-Load Equipment. Berkeley, CA: Lawrence Berkeley National Laboratory.
Roberts, Simon, and William Baker. 2003. Towards Effective Energy Information: Improving Consumer Feedback on Energy Consumption. Bristol, UK: Centre for Sustainable Energy.
Schultz, P. Wesley, Mica Estrada, Joseph Schmitt, Rebecca Sokoloski, and Nilmini Silva-Send. 2015. "Using in-home displays to provide smart meter feedback about household electricity consumption: A randomized control trial comparing kilowatts, cost, and social norms." *Energy* 90, Part 1:351-358. doi: http://dx.doi.org/10.1016/j.energy.2015.06.130.
Schultz, P. Wesley, Jessica M. Nolan, Robert B. Cialdini, Noah J. Goldstein, and Vladas Griskevicius. 2007. "The Constructive, Destructive, and Reconstructive Power of Social Norms." *Psychological Science* 18 (5):429-434. doi: https://doi.org/10.1111/j.1467-9280.2007.01917.x.
Schwarz, Norbert, Hans-J. Hippler, Brigitte Deutsch, and Fritz Strack. 1985. "Response Scales: Effects of Category Range on Reported Behavior and Comparative Judgments." *Public Opinion Quarterly* 49 (3):388-395.
Spence, A., C. Leygue, B. Bedwell, and C. O'Malley. 2014. "Engaging with energy reduction: Does a climate change frame have the potential for achieving broader sustainable behaviour?" *Journal of Environmental Psychology* 38:17-28. doi: https://doi.org/10.1016/j.jenvp.2013.12.006.
Steinhorst, Julia, and Christian A. Klöckner. 2017. "Effects of Monetary Versus Environmental Information Framing: Implications for Long-Term Pro-Environmental Behavior and Intrinsic Motivation." *Environment and Behavior* 0 (0):001391651772537. doi: 10.1177/0013916517725371.
Tiedemann, K., I. Sulyma, and E. Mazzi. 2013. "Residential Behavioral Savings: An Analysis of Principal Electricity End Uses in British Columbia." BECC Conference, Sacramento, CA.
UEFI Forum. 2018. "Specifications." accessed November 19, 2018. http://uefi.org/specifications.
United States Environmental Protection Agency. 2018. "Learn About FEC." accessed November 19, 2018. https://www.epa.gov/fec/learn-about-fec.
Urban, Bryan, Kurt Roth, Mahendra Singh, and Duncan Howes. 2017. Energy Consumption of Consumer Electronics in U.S. Homes in 2017. Boston, MA: Fraunhofer USA Center for Sustainable Energy Systems.
Vine, D., L. Buys, and P. Morris. 2013. "The effectiveness of energy feedback for conservation and peak demand: A literature review." *Open Journal of Energy Efficiency* 2 (1):7-15.
Yun, Ray, Azizan Aziz, Bertrand Lasternas, Vivian Loftness, Peter Scupelli, and Chenlu Zhang. 2017. "The persistent effectiveness of online feedback and controls for sustainability in the workplace." *Energy Efficiency* 10 (5):1143-1153. doi: http://dx.doi.org/10.1007/s12053-017-9509-4.

APPENDIX A:
Frequently Asked Questions Page Text

This appendix reproduces the text of the Frequently Asked Questions page provided in the PMUI application.

Frequently Asked Questions

Need more help? Contact us at pmui@calit2.uci.edu

What does PMUI stand for?

Power Management User Interface. It's not a very catchy name, we admit. We're open to suggestions for the next version.

What do the terms "off," "sleep," "active," and "idle" mean?

PMUI measures your computer at three states: off, sleep, and on (whether active or idle). Any other states your computer may go into are coded as one of these, based on using the same energy consumption. For example, hibernate uses so little power that it is coded as "off," and hybrid sleep is coded as "sleep." When your computer is "on," the PMUI program also records whether it is "active" or "idle." The computer is "active" when you are using it, by typing or moving the mouse or doing something like watching a video, and idle if you are not using it. If you are operating the computer via remote access, PMUI codes your activity the same way. When you use the computer, it's natural to have short periods when you're not touching it (for instance, while reading), so we wait for 10 minutes of no activity before we code the computer as being "idle."

How much energy does my computer use in each of these states?

Computers vary in how much energy they use while on, with newer computers generally being more efficient, and high performance computers and older models usually using more energy. In general, computers use almost as much energy when idle – that is, left on when not being used – as when active. Sleep mode uses as little as 5% of the energy as on mode, and computers use almost no energy when off. A small amount of idle time is necessary for your computer to run processes in the background. However, for most computers, the majority of time they spend idle is wasted energy, because the computer could be saving energy in "sleep" mode or "off."

How can I save more energy with my computer?

The more idle time you reduce, the more energy you save. The most reliable way to do this is to enable your computer's automatic sleep settings and set the delay for a short period, such as 20 minutes. (See the "Sleep Settings" screen.) You can also manually put your computer to sleep as soon as you're done with it, and turn it off at night and on the weekends. Although turning the computer off saves the most energy, it's easy to forget and

accidentally leave the computer on all weekend, which is why it's important to have your sleep settings enabled.

What about the energy used by my computer display/monitor?

Displays/monitors use less energy than desktop computers, although even that still adds up over time. However, most people already have their display sleep settings set to 10 or 15 minutes, so the PMUI feedback reports focus on computer idle time instead. Screensaver programs do not save any energy at all for the monitor/display; sleep mode should be used instead. Your display uses almost as little energy when in sleep mode as when off.

What is the Energy Saver Target Profile?

This shows the percent of idle time that would be ideal for the level of active time shown by your computer. The target can be attained by setting a low delay time for computer sleep, consistently putting your computer to sleep or turning it off when not in use, or a combination of the two. The target profile does not designate an ideal proportion of off versus sleep, as users differ in whether they can or will turn their computers off. Although computers use less energy when off than when asleep, the most important step for saving energy is to reduce idle time.

Why is some of the data missing on my report pages?

For any chart that shows a percentage for a day, week, or month, the chart cannot be generated until the program has all the data for that entire day, week, or month. So you cannot see today's results until tomorrow. Also, you will not be able to see a percent of time spent in each state per week until the end of your first week. If data for an already-completed time interval is missing, please contact us.

Why do the reports say my computer went off if I didn't turn it off?

Very brief off periods (usually less than a minute) are most often caused by the computer being rebooted, such as for software upgrades. If your computer is reported off for longer periods, your computer may be set to automatically transition to hibernate mode after a time delay, and PMUI records hibernate as being off. Hibernate mode is like a deeper version of sleep, that saves the files you had open, but uses almost as little energy as turning your computer off.

Why do the reports say my computer was on when it should be in sleep mode?

Some computers turn on to perform processes, such as automatic software upgrades or backups, frequently in the middle of the night. If your computer's automatic sleep settings are enabled, the computer should go back to sleep when these processes are finished.

If I change my settings on PMUI, do I have to change them on my standard settings too?

No, don't worry about it. Any changes you make to PMUI will be automatically changed in your regular power management settings too, and any changes you make to those settings will show up in PMUI.

What is the “temporarily disable sleep settings” feature for?

Users sometimes disable their sleep settings to make sure their computer stays on during a certain process, but then forget to re-enable the settings. With this new feature, you don’t have to remember to re-enable: whatever settings you had before will automatically resume after the time period you specify. If you finish your task earlier than expected, you can uncheck the blue “Temporarily Disable Sleep Settings” box to immediately re-enable the settings.

Can I remotely access my desktop if it’s asleep?

Technically no, but you may be able to activate “Wake-on-LAN” (WoL) protocols that will wake your desktop so you can remotely access it. If you’re an advanced user, you can search online for instructions for your specific operating system. Otherwise, we suggest getting help from an expert, such as your IT manager, as making the required changes can damage your computer if done improperly.

Can I disable the weekly energy reminder popups?

Yes, you can -- uncheck this box. Weekly popups: [ ]

Can I uninstall PMUI?

Yes, you can uninstall it the same way as any other programs on your computer. If you don’t know how to do that, try searching online for “how to uninstall programs” for your particular operating system.

APPENDIX B:
Software Resources

This section lists the software resources used in the development process, which are all free or open-source software.

Python Libraries Used

- CGI: CGI script is invoked by an HTTP server, usually to process user input submitted through an HTML <FORM> or <ISINDEX> element
- Codecs: Defines a base classes for standard Python codecs
- Io: Provides Python's main facilities for dealing with various types of I/O
- Json: A lightweight data interchange format inspired by JavaScript object literal syntax
- Logging: Defines functions and classes which implement a flexible event logging system for applications and libraries
- Optparse: Convenient, flexible, and powerful library for parsing command-line options than the old getopt module
- Os: Provides a portable way of using operating system dependent functionality
- Os.path: implements some useful functions on pathnames
- Posixpath: A subclass of Pure Path, this path flavour represents non-Windows file system paths
- Re: regular expressions
- Shelx: Shlex class makes it easy to write lexical analyzers for simple syntaxes resembling that of the unix shell
- SimpleHTTPServer: Defines a single class, SimpleHTTPRequestHandler, which is interface-compatible with BaseHTTPServer
- SocketServer: Simplifies the task of writing network servers
- Subprocess: Allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes
- Sys: Provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter
- Tempfile: Creates temporary files and directories
- Unittest: Unit testing framework to help test your code
- Urllib: A package that collects several modules for working with URLs
- xml.etree.ElementTree: implements a simple and efficient API for parsing and creating XML data

JavaScript Libraries Used

- Abbrev: calculate the set of unique abbreviations for a given set of strings
- Accessibility-developer-tools: A library of accessibility-related testing and utility code
- Acorn: ECMAScript parser
- Align-text: Align test values in an array
- Alphanum-sort: Alphanumeric sorting algorithm
- Amdefine: Provide AMD's define() API for declaring modules in the AMD format
- Ansi-regex: Regular expression for matching ANSI escape codes
- Ansi-styles: ANSI escape codes for styling strings in the terminal

- Anymatch: Matches strings against configurable strings, globs, regular expressions, and/or functions
- Apple-Syslog-Stream: Provides a stream parser for Apple Syslog
- Apple-system-listener: Reports the following events: sleep, wake, shutdown, boot, power settings change, user active, and user idle
- Apple-system-profiler: Wrapper around the apple's system_profiler that parses the output
- Aproba: A ridiculously light-weight argument validator
- Archiver: A streaming interface for archive generation
- Archiver-utils: Utility functions for archiver
- Are-we-there-yet: Keep track of the overall completion of many disparate processes
- Argparse: Very powerful CLI arguments parser. Native port of argparse - pythons options parsing library
- Array-find-index: Finds the index in an array
- Array-unique: Remove duplicate values from an array. Fastest ES5 implementation
- Arr-diff: Returns an array with only the unique values from the first array, by excluding all values from additional arrays using strict equality for comparisons
- Arr-flatten: Recursively flatten an array of arrays. This is the fastest implementation of array flatten
- Arrify: Convert a value to an array
- Asap: High-priority task queue for node.js and browsers
- Asar: Creating Electron app packages
- Asn1: Contains parsers and serializers for ASN.1
- Assert: Used for writing unit tests for your applications, you can access it with require("assert")
- Assertion-error: Error constructor for test and validation frameworks that implements standardized AssertionError specification
- Assert-plus: Extra assertions on top of node's assert module
- Async: Higher-order functions and common patterns for asynchronous code
- Async-each: async parallel forEach / map function for javascript
- Autoprefixer: Parse CSS and add vendor prefixes to CSS rules using values from the Can I Use website
- Aws4: Signs and prepares requests using AWS Signature Version 4
- Aws-sign2: AWS Signing. Originally pulled from LearnBoost/knox, maintained as vendor in request, now a standalone module
- Axios: Promise based HTTP client for the browser and node.js
- Babel: Turn ES6 code into readable vanilla ES5 with source maps
- Babel-code-frame: Generate errors that contain a code frame that point to source locations
- Babel-core: babel compiler core
- Babel-generator: turns an AST into code
- Babel-helper-builder-react-jsx: Helper function to build react jsx
- Babel-helper-call-delegate: helper function to call delegate
- Babel-helper-define-map: Helper function to define a map
- Babel-helper-function-name: Helper function to change the propery 'name' of every function
- Babel-helper-get-function-arity: Helper function to get arity
- Babel-helper-get-function-arity: Helper function to get function arity
- Babel-helper-hoist-variables: Helper function to hoist variables
- Babel-helper-optimise-call-expression: Helper function to optimise call expression
- Babel-helper-regex: Helper function to check for literal RegEx
- Babel-helper-replace-supers: Helper function to replace supers
- Babel-helpers: collection of helper functions used by babel transforms

- Babel-loader: Babel module loader for webpack
- Babel-messages: collection of debug messages used by babel
- Babel-plugin-check-es2015-constants: Compile ES2015 constants to ES5
- Babel-plugin-detective: Babel % plugin that scans the AST for require calls and import statements
- Babel-plugin-syntax-async-functions: Allow parsing of async functions
- Babel-plugin-syntax-flow: Allow parsing of the flow syntax
- Babel-plugin-syntax-jsx: Allow parsing of jsx
- Babel-plugin-syntax-object-rest-spread: Allows parsing of object rest/spread
- Babel-plugin-transform-es2015-arrow-functions: Compile ES2015 arrow functions to ES5
- Babel-plugin-transform-es2015-block-scoped-functions: Babel plugin to ensure function declarations at the block level are block scoped
- Babel-plugin-transform-es2015-block-scoping: Compile ES2015 block scoping (const and let) to ES5
- Babel-plugin-transform-es2015-classes: Compile ES2015 classes to ES5
- Babel-plugin-transform-es2015-computed-properties: Compile ES2015 computed properties to ES5
- Babel-plugin-transform-es2015-destructuring: Compile ES2015 destructuring to ES5
- Babel-plugin-transform-es2015-duplicate-keys: Compile objects with duplicate keys to valid strict ES5
- Babel-plugin-transform-es2015-for-of: Compile ES2015 for ...of to ES5
- Babel-plugin-transform-es2015-function-name: Apply ES2015 function.name semantics to all functions
- Babel-plugin-transform-es2015-literals: Compile ES2015 unicode string and number literals to ES5
- Babel-plugin-transform-es2015-modules-amd: Transforms ES2015 modules to AMD
- Babel-plugin-transform-es2015-modules-commonjs: plugin transforms ES2015 modules to commonJS
- Babel-plugin-transform-es2015-modules-systemjs: Plugin transforms ES2015 modules to SystemJS
- Babel-plugin-transform-es2015-modules-umd: Transforms ES2015 modules to UMD
- Babel-plugin-transform-es2015-object-super: Compile ES2015 object super to ES5
- Babel-plugin-transform-es2015-parameters: Compile ES2015 default and rest parameters to ES5
- Babel-plugin-transform-es2015-shorthand-properties: Compile ES2015 shorthand properties to ES5
- Babel-plugin-transform-es2015-spread: Compile ES2015 spread to ES5
- Babel-plugin-transform-es2015-sticky-regex: Compile ES2015 sticky regex to an ES5 RegExp constructor
- Babel-plugin-transform-es2015-template-literals: Compile ES2015 template for literals to ES5
- Babel-plugin-transform-es2015-typeof-symbol: This transformer wraps all typeof expressions with a method that replicates native behavior
- Babel-plugin-transform-es2015-unicode-regex: Compile ES2015 Unicode regex to ES5
- Babel-plugin-transform-flow-strip-types: Strip flow type annotations from your output code
- Babel-plugin-transform-object-rest-spread: Compile object rest and spread to ES5
- Babel-plugin-transform-react-display-name: Add displayName to React.createClass calls
- Babel-plugin-transform-react-jsx: Turn JSX into react function calls
- Babel-plugin-transform-react-jsx-self: Add a __self prop to all JSX Elements
- Babel-plugin-transform-react-jsx-source: Add a __source prop to all JSX Elements
- Babel-plugin-transform-regenerator: Explode async and generator functions into a state machine

- Babel-plugin-transform-strict-mode: places a 'use strict' directive at the top of all files to enable strict mode
- Babel-preset-es2015: Babel presets for all ES2015 plugins
- Babel-preset-react: Babel preset for all react plugins
- Babel-register: babel require hook
- Babel-runtime: babel selfContained runtime
- Babel-template: generate an AST from a string template
- Babel-Traverse: maintains the overall tree state, and is responsible for replacing, removing, and adding nodes
- Babel-types: babel types is a lodash-esque utility library for AST nodes
- Babylon: a javascript parser
- Balanced-match: Match balanced character pairs
- Base64: A C++ module for node-js that does base64 encoding and decoding
- Base64-js: Base64 encoding/decoding in pure JS
- Big.js: A small, fast, easy-to-use library for arbitrary-precision decimal arithmetic
- Binary: Unpack multibyte binary values from buffers
- Binary-extensions: List of binary file extensions
- Bl: Pure javascript Biginteger module for node.js
- Block-stream: A stream of blocks
- Bluebird: a full featured promise library with unmatched performance
- Boom: HTTP-friendly error objects
- Brace-expansion: Brace expansion as known from sh/bash
- Braces: Fast, comprehensive, bash-like brace expansion implemented in JavaScript.
- Browserify-zlib: Full zlib module for browserify
- Browserslist: Get browser versions that matches given criterias like an Autoprefixer
- Buffer: makes it possible to interact with octet streams in the context of things like TCP streams and file system operations
- Buffer-crc32: Pure JavaScript CRC32 algorithm that plays nice with binary data
- Bufferlist: Create linked lists of buffer objects
- Buffers: Treat a collection of buffers as a single contiguous partially mutable buffer
- Buffer-shims: Some shims for node buffers
- Builtin-modules: List of the Node.js builtin modules
- Camelcase: Convert a dash/dot/underscore/space separated string to camelCase: foo-bar -> fooBar
- Camelcase-keys: Convert object keys to camelCase
- Caniuse-db: Raw browser/feature support data from caniuse.com
- Caseless: Caseless object set/get/has, very useful when working with HTTP headers
- Center-align: Center-align the text in a string
- Chai: Assertion library for node and the browser that can be delightfully paired with any javascript testing framework
- Chainsaw: Build chainable fluent interfaces the easy way
- Chalk: terminal string styling
- Chokidar: A neat wrapper around node.js fs.watch / fs.watchFile / fsevents
- Chromium-pickle-js: Binary value packing and unpacking
- Clap: Command line argument parser
- Classnames: Simple utility for conditionally joining classNames together
- Cliui: Easily create complex multi-column command-line-interfaces
- Clone: Deep cloning of objects and arrays
- Coa: Command-option-argument: Yet another parser for command line options
- Code-point-at: returns a non-negative integer that is the Unicode code point value
- Color: Color conversion and manipulation with CSS string support
- Color-convert: Plain color conversion functions
- Colormin: Turn a CSS color into its smallest representation

- Color-name: A list of color names and it's values
- Colors: Get colors in your node.js console
- Color-string: Parser and generator for CSS color strings
- Combined-stream: A stream that emits multiple other streams one after another
- Commander: The complete solution for node.js command-line programs
- Commondir: Compute the closest common parent for file paths
- Compress-commons: A library that defines a common interface for working with archive formats within node
- Concat-map: Concatenative map dashery
- Concat-stream: Writable stream that concatenates strings or binary data and calls a callback with the result
- Console-browserify: Emulate console for all the browsers
- Console-control-strings: Library of cross-platform tested animal/console command strings for doing things like color and cursor positioning.
- Constants-browserify: Node's constants module for the browser
- Convert-source-map: converts a source-map from/to different formats and allows adding/changing properties
- Core-js: Modular standard library, includes polyfills for ECMAScript 5, ECMAScript 6: promises, symbols, collections, etc.
- Core-util-is: A testing library to assert objects and their values
- Crc32-stream: Streaming CRC32 checksum
- Cron: Cron jobs for your node
- Cross-spawn: Cross platform child_process#spawn and child_process#spawnSync
- Cryptiles: General purpose crypto utilities
- Crypto : provides cryptographic functionality that includes a set of wrappers for OpenSSL's hash, HMAC, cipher, decipher, sign and verify functions
- Crypto-browserify: Implementation of crypto for the browser
- Css-color-names: JSON Object of css color names mapped to their hex value
- Cssec: Library for escaping CSS strings and identifiers while generating the shortest possible ASCII-only output
- Css-loader: css loader module for webpack
- Cssnano: Modular minifier, built on top of the PostCSS ecosystem
- Csso: CSS Optimizer is a CSS minifier with structural optimizations
- Css-selector-tokenizer: Parse and stringify CSS selectors
- Ctype: Read and write binary structures and data types
- Cuint: Unsigned integers for javascript
- Currently-unhandled: Track the list of currently unhandled promise rejections
- D3: Visualizing data using web standards
- D3-array: Array manipulation, ordering, searching, summarizing, etc.
- D3-axis: Displays automatic reference lines for scales
- D3-brush: Select a one- or two-dimensional region using the mouse or touch
- D3-chord: Visualize relationships or network flow with an aesthetically-pleasing circular layout
- D3-collection: Handy data structures for elements keyed by string
- D3-color: Color spaces, RGB< Cubehelix, Lab and HCL
- D3-dispatch: Register named callbacks and call them with arguments
- D3-drag: Drag and drop SVG, HTML or Canvas using mouse or touch input
- D3-dsv: Parser and formatter for delimiter-separated values, such as CSV and TSV
- D3-ease: Easing functions for smooth animation
- D3-force: Force-directed graph layout using velocity Verlet integration
- D3-format: Format numbers for human consumption
- D3-geo: Shapes and calculators for spherical coordinates
- D3-hierarchy: Layout algorithms for visualizing hierarchical data

- D3-interpolate: Interpolate numbers, colors, strings, arrays, objects, whatever! Blends two values, values may be numbers, colors, strings, arrays, or even deeply-nested objects
- D3-path: Serialize canvas path commands to SVG
- D3-polygon: Operations for two-dimensional polygons
- D3-quadtree: 2-d recursive spatial subdivision
- D3-queue: Evaluate asynchronous tasks with configurable concurrency
- D3-random: Generate random numbers from various distributions
- D3-request: A convenient alternative to XMLHttpRequest
- D3-scale: Encodings that map abstract data to visual representation
- D3-selection: Data-driven DOM manipulation: select elements and join them to data
- D3-shape: Graphical primitives for visualization, such as lines and areas
- D3-time: A calculator for humanity's peculiar conventions of time
- D3-time-format: Calculator for humanity's peculiar conventions of time
- D3-timer: Efficient queue capable of managing thousands of concurrent animations
- D3-transition: Animated transitions for D3 selections
- D3-voronoi: Compute the Voronoi diagram of a set of two-dimensional points
- D3-zoom: Pan and zoom SVG, HTML or Canvas using mouse or touch input
- Dashdash: A light, featureful and explicit option parsing library
- Date-now: A requirable version of Date.now()
- Debounce: Creates and returns a new debounced version of the passed function that will postpone its execution until after x milliseconds have elapsed since the last time it was invoked
- Debug: a small debugging utility
- Decamelize: Convert camelized string into a lowercased one with a custom separator
- Decompress-zip: Extract files from a ZIP Archive
- Deep-eql: Improved deep equality testing for Node.js and the browser
- Deep-equal: Node's assert.deepEqual algorithm
- Deep-extend: Recursive object extending
- Defined: Return the first argument that is !== undefined
- Delayed-stream: Buffers events from a stream until you are ready to handle them
- Delegates: Delegate methods and accessors to another property
- Detect-incident: Detect the indentation of code
- Devtron: Electron DevTools extension
- Diff: JavaScript text diff implementation
- Domain-browser: Node's domain module for the web browser. This is merely an evented try...catch with the same API as node.
- Duplex: Base class for a duplex stream
- Electron: Install prebuilt electron binaries for the command-line using npm
- Electron-debug: Adds useful debug features to your electron app
- Electron-dev: Wraps your program, however where node-dev actually hooks into require(), electron-dev uses babel-plugin-detective to acquire a list of all the js files
- Electron-download: Download electron prebuilt binary zips from github releases
- Electron-is-dev: Check if Electron is running in development
- Electron-localshortcut: Register/unregister a keyboard shortcut locally to a BrowserWindow instance, without using a menu
- Electron-packager: Package and distribute your Electron app with Os-specific bundles
- Electron-prebuilt: Install prebuilt electron binaries for the command-line using npm
- Electron-sudo: Electron subprocesses with administrative privileges, prompting the user with an OS dialog if necessary
- Electron-winstaller: Module to generate Windows installers for Electron Apps
- Emojis-list: Complete list of standard emojis
- Encoding: Convert encodings, uses iconv by default and fallbacks to iconv-lite if needed

- End-of-stream: Call a callback when a readable/writable/duplex stream has completed or failed
- Enhanced-resolve: Offers a async require.resolve function. Highly configurable
- Errno: Libuv errno details exposed
- Error-ex: Easy error subclassing and stack customization
- Escape-string-regexp: Escape RegExp special characters
- Esprima: ECMAScript parsing infrastructure for multipurpose analysis
- Esutils: Utility box for ECMAScript language tools
- Events: Node's event emitter for all engines
- Events.EventEmitter: Node's event emitter for all engines
- Expand-range: Fast, bash-like range expansion.
- Extend: Port of jQuery.extend for node.js and the browser
- Extglob: Extended glob support for JS. Adds the expressive power of regular expression to glob patterns
- Extract-zip: Unzip a zip file into a directory using 100% pure gluten-free organic javascript
- Extsprintf: Extended POSIX-style sprintf
- Fastparse: Simple and stupid parser, based on a state machine and regular expressions
- Fbjs: Collection of utility libraries used by other Facebook JS projects
- Fd-slicer: Safely create multiple ReadStream or WriteStream objects from the same file descriptor
- File-loader: File loader module for webpack
- Filename-regex: Regular expression for matching file names, with or without extension
- Fill-range: Fill in a range of numbers or letters, optionally an increment or step to use, of create a regex-compatible range
- Find-cache-dir: Finds the common standard cache directory
- Find-up: Find a file by walking up parent directories
- Flatten: Flatten arbitrarily nested arrays into a non-nested list of non-array items
- Follow-redirects: HTTP and HTTPS modules that follow redirects
- Font-awesome: Iconic font and CSS framework
- Forever-agent: HTTP Agent that keeps socket connections alive between keep-alive requests
- For-in: Iterate over the own and inherited enumerable properties of an object, and return an object with properties that evaluate to true from the callback
- Form-data: Create readable "multipart/form-data" streams
- For-own: Iterate over the own enumerable properties of an object, and return an object with properties that evaluate to true from the callback
- Fs: File I/O is provided by simple wrappers around standard POSIX functions
- Fs.realpath: Use node's fs.realpath, but fall back to the JS implementation if the native one fails
- Fsevents: Native Access to Mac OS-X FSEvents
- Fs-extra: Contains methods that aren't included in the vanilla Node.js fs package
- Fstream: Advanced file system stream things
- Fstream-ignore: A thing for ignoring files based on globs
- Generate-function: Module that helps you write generated functions in Node
- Generate-object-property: Generate safe JS code that can be used to reference an object property
- Get-package-info: Gets properties from package.json files in parent directories
- Getpass: Get a password from the terminal
- Get-stdin: Get stdin as a string or buffer
- Glob: Match files using the patterns the shell uses
- Globals: Global identifiers from different JavaScript environments
- Glob-base: Returns an object with the base path and the actual pattern

- Glob-Parent: strips glob magic from a string to provide the parent directory path
- Graceful-fs: A drop-in replacement for fs, making various improvements
- Graceful-readlink: graceful fs.readlink
- Growl: Growl unobtrusive notifications
- Guage: A nearly stateless terminal based horizontal gauge/progress bar
- Har-validator: Extremely fast HTTP archive (HAR) validator using JSON Schema
- Has-ansi: Check if a string has ANSI escape codes
- Has-color: Detect whether a terminal supports color
- Has-flag: Check if argv has a specific flag
- Has-own: A safer .hasOwnProperty() -hasOwn(name, obj)
- Has-unicode: Try to guess if your terminal supports unicode
- Hawk: HTTP Hawk Authentication Scheme
- Highlight: Highlight code syntax with node.js
- History: Manage session history with JavaScript
- Hoek: General purpose node utilities
- Hoist-non-react-statics: Copies non-react specific statics from a child component to a parent component
- Home-or-tmp: Get the user home directory with fallback to the system temp directory
- Home-path: Cross-platform home directory retriever
- Hosted-git-info: provides metadata and conversions from repository urls for github, bitbucket, and gitlab
- Html-comment-regex: Regular expression for matching HTML comments
- Http-browserify: HTTP module compatibility for browserify
- Https-browserify: HTTPS module compatibility for browserify
- Http-signature: Reference implementation of Joyent's HTTP Signature scheme
- Humanize-plus: A simple utility library for making the web more humane
- Iconv-lite: Convert character encodings in pure javascript
- Icss-replace-symbols:
- Ieee754: Read/write IEEE754 floating point numbers from/to a Buffer or array-like object
- Image-size: Get dimensions of any image file
- Immutable: Immutable Data Collections
- Indent-string: Indent each line in a string
- Indexes-of: Line string/array#indexOf but return all the indexes in an array
- Indexof: Retrieves the index of
- Inflight: Add callbacks to requests in flight to avoid async duplication
- Inherits: Browser-friendly inheritance fully compatible with standard node.js inherits()
- Ini: An ini encoder/decoder for node
- Interpret: A dictionary of file extensions and associated module loaders
- Invariant: a way to provide descriptive errors in development but generic errors in production
- Is-absolute-url: Check if a URL is absolute
- Isarray: For older browsers (isArray)
- Is-arrayish: Determines if an object can be used as an array
- Is-binary-path: Check if a filepath is a binary file
- Is-buffer: Determine if an object is a buffer
- Is-bulletin-module: Check if a string matches the name of a Node.js builtin module
- Is-dotfile: Return true if a file path is(or has) a dotfile. Returns false if the path is a dot directory
- Is-equal-shallow: Does a shallow comparison of two objects, returning false if the keys or values differ
- Isexe: Minimal module to check if a file is executable

- Is-extendable: Returns true if a value is any of the object types: array, regexp, plain object, function or date
- Is-extglob: Returns true if a string has an extglob
- Is-finite: Returns a boolean value if a number is finite or not
- Is-fullwidth-code-point: Check if the character represented by a given Unicode point is fullwidth
- Is-glob: Returns true if the given string looks like a glob pattern or an extglob pattern
- Is-my-json-valid: A JSONSchema validator that uses code generation to be extremely fast
- Is-number: Returns true if the value is a number
- Isobject: Checks whether a value is an object
- Isomorphic-fetch: Fetch for node and browserify
- Is-plain-obj: Check if a value is a plain object
- Is-posix-bracket: Returns true if the given string is a POSIX bracket expression
- Is-primitive: Returns true if the value is a primitive
- Is-promise: Returns true if the value is a promise
- Is-property: Tests if a JSON property can be accessed using . syntax
- Is-stream: Check if something is a Node.js stream
- Isstream: Determine if an object is a Stream
- Is-svg: Check if a string or buffer is SVG
- Is-typedarray: Detect whether or not an Object is a Typed Array
- Is-utf8: Detect if a buffer is utf8 encoded
- Jade: A clean, white space-sensitive template language for writing HTML
- Jodid25519: Curve 25519-based cryptography
- Js-base64: Yet another Base64 transcoder in pure-JS
- Jsbn: Fast, portable implementation of large-number math in pure JavaScript
- Jsesc: Secure your JSON
- Json5: an excellent data format, but done better
- Jsonfile: Easily read/write JSON files
- Json-loader: JSON loader module for webpack
- Jsonparse: This is a pure-js JSON streaming parser for Node.js
- Jsonpointer: Simple JSON addressing
- Json-schema: JSON Schema validation and specifications
- JSONStream: Streaming JSON.parse and stringify
- Json-stringify-safe: Like JSON.stringify, but doesn't blow up on circular refs
- Jsprim: Utilities for primitive JS types
- Js-tokens: A regex that tokenizes JavaScript
- Js-yaml: YAML 1.2 parser and serializer
- Kind-Of: Get the native type of a value
- Klaw: File system walker with Readable stream interface
- Lazy-cache: Cache requires to be lazy-loaded when needed
- Lazystream: Open node streams on demand
- Less: Learner CSS
- Less-loader: Less loader module for webpack
- Loader-utils: utils for webpack loaders
- Load-json-file: Read and parse a JSON file
- Lodas._root: Internal lodash function root exported as a Node.js module
- Lodash: Lodash modular utilities
- Lodash._createcompounder: Modern build of lodash's internal createCompounder exported as a Node.js/io.js module
- Lodash.camelcase: Modern build of lodash's_.camelCase exported as a Node.js/io.js module
- Lodash.deburr: Modern build of lodash's_.deburr exported as a Node.js/io.js module
- Lodash.get: Gets the value at path of object

- Lodash.template: Modern build of lodash's_.template exported as a Node.js/io.js module
- Lodash.templatesettings: Modern build of lodash's_.templateSettings exported as a Node.js/io.js module
- Lodash.words: Modern build of lodash's_.words exported as a Node.js/io.js module
- Lodash-es: Lodash exported as ES modules
- Longest: Get the longest item in an array
- Loose-envify: Fast (and loose) selective process.env replacer using js-tokens instead of an AST
- Loud-rejection: Make unhandled promise rejections fail loudly instead of the default silent fail
- Lovefield: provides SQL-like syntax and works cross-browser
- Lru-cache: A cache object that deletes the least-recently-used items
- Macaddress: Get the MAC addresses (hardware addresses) of the host's network interfaces
- Map-obj: Map object keys and values into a new object
- Memory-fs: A simple in-memory filesystem, holds data in a javascript object
- Meow: CLI app helper
- Micromatch: Glob matching for javascript/node.js. A drop-in replacement and faster alternative to minimatch and multimatch
- Mime: A comprehensive library for mime-type mapping
- Mime-db: Media Type Database
- Mime-types: The ultimate JavaScript content-type utility
- Minimatch: a glob matcher in javascript
- Minimist: Parse argument options
- Mkdirp: Create a new directory and any necessary subdirectories at dir with octal permission string opts.mode.
- Mkpath: Make all the directories in a path, like mkdir-p
- Mksnapshot: Create a snapshot file for electron
- Mocha: Simple, flexible, fun test framework
- Moment: Parse, validate, manipulate, and display dates
- Moment-timezone: Parse and display moments in any timezone
- Mousetrap: Simple library for handling keyboard shortcuts
- Ms: Tiny millisecond conversion utility
- Mv: fs.rename but works across devices. Same as the unix utility 'mv'
- Nan: Native abstractions for Node.js: C++ header for node 0.8 -> 6 compatibility
- Ncp: Asynchronous recursive file copy utility
- Node-fetch: A lightweight module that brings window.fetch to node.js and io.js
- Node-int64: Support for representing 64-bit integers in JavaScript
- Node-libs-browser: The node core libs for in browser usage
- Node-uuid: Rigorous implementation of RFC4122 (v1 and v4) UUIDs
- Nopt: Option parsing for Node, supporting types, shorthands, etc. Used by npm
- Normalize-package-data: Normalizes data that can be found in package.json files
- Normalize-path: slashes to be unix-like forward slashes. Also condenses repeat slashes to a single slash and removes trailing slashes
- Normalize-range: Utility for normalizing a numeric range, with a wrapping function useful for polar coordinates
- Normalize-url: normalize a URL, useful when you need to display, store, deduplicate, sort, compare, etc, URLs
- Nugget: Minimalist wget clone written in node. HTTP GETs a file and saves it to the current working directory
- Num2fraction: convert number to fraction
- Number-is-nan: A Polyfill that doesn't overwrite the native method

- Oauth-sign: OAuth 1 signing. Formerly a vendor lib in mikeal/request, now a standalone module
- Object.omit: Return a copy of an object excluding the given key, or array of keys. Also accepts an optional filter function as the last argument
- Object-assign: A polyfill that doesn't overwrite the native method
- Object-keys: An Object.keys replacement. In case Object.keys is not available
- Once: Run a function exactly one time
- Optimist: Light-weight option parsing with an argv hash. No optstrings attached
- Os: provides a number of operating system-related utility methods
- Os-browserify: OS module from Node.js, but for browsers
- Os-homedir: Node.js os.homedir() polyfill
- Os-tmpdir: Node.js os.tmpdir() polyfill
- Pako: Zlib port to javascript - fast, modularized, with browser support
- Parse-glob: Parse a glob pattern into an object of tokens
- Parse-json: Parse JSON with more helpful errors
- Path: provides utilities for working with file and directory paths
- Path-browserify: The path module from node core for browsers
- Path-exists: Check if a path exists
- Path-is-absolute: determines if path is an absolute path
- Path-type: Check if a path is a file, directory, or symlink
- Pbkdf2-compat: Provides the functionality of PBKDF2 with the ability to use any supported hashing algorithm returned from crypto.getHashes()
- Pend: Dead-simple optimistic async helper
- Pify: Promisify a callback-style function
- Pinkie: This module is an exact Promise specification polyfill (like native-promise-only), but in Node.js land (it should be browserify-able though)
- Pinkie-promise: ES2015 Promise polyfill
- Pkg-dir: Find the root directory of a npm package
- Plist: Mac OS X Plist parser/builder for Node.js and browsers
- Postcss: Tool for transforming styles with JS plugins
- Postcss-calc: Plugin reduce calc() references whenever it's possible
- Postcss-colormin: Minify colors in your CSS files with PostCSS
- Postcss-convert-values: Convert values with PostCSS
- Postcss-discard-comments: Discard comments in your CSS files with PostCSS
- Postcss-discard-duplicates: Discard duplicate rules in your CSS files with PostCSS
- Postcss-discard-empty: Discard empty rules and values with PostCSS
- Postcss-discard-overridden: PostCSS plugin to discard overridden @keyframes or @counter-style
- Postcss-discard-unused: Discard unused counter styles, keyframes, and fonts
- Postcss-filter-plugins: Exclude/warm on duplicated PostCSS plugins
- Postcss-merge-idents: Merge keyframe and counter style identifiers
- Postcss-merge-longhand: Merge longhand properties into shorthand with PostCSS
- Postcss-merge-rules: Merge CSS rules with PostCSS
- Postcss-message-helpers: PostCSS helpers to throw or output GNU style messages
- Postcss-minify-font-values: Minify font declarations with PostCSS
- Postcss-minify-gradients: Minify gradient parameters with PostCSS
- Postcss-minify-params: Minify at-rule params with PostCSS
- Postcss-minify-selectors: Minify selectors with PostCSS
- Postcss-modules-extract-imports: A CSS modules transform to extract local aliases for inline imports
- Postcss-modules-local-by-default: A CSS modules transform to make local scope to the default

- Postcss-modules-scope: A CSS Modules transform to extract export statements from local-scope classes
- Postcss-modules-values: PostCSS plugin for CSS Modules to pass arbitrary constants between your module files
- Postcss-normalize-charset: Add necessary or remove extra charset with PostCSS
- Postcss-normalize-url: Normalize URLs with PostCSS
- Postcss-ordered-values: Ensure values are ordered consistently in your CSS
- Postcss-reduce-idents: Reduce custom identifiers with PostCSS
- Postcss-reduce-initial: Reduce initial definitions to the actual initial value, where possible
- Postcss-reduce-transforms: Reduce transform functions with PostCSS
- Postcss-selector-parser: Selector parser with built in methods for working with selector strings
- Postcss-svgo: Optimize inline SVG with PostCSS
- Postcss-unique-selectors: Ensure CSS selectors are unique
- Postcss-value-parser: transforms css values and at-rule params into the tree
- Postcss-zindex: Reduce z-index values with PostCSS
- Prepend-http: Prepend http:// to humanize URLs like todomvc.com and localhost
- Preserve: Temporarily substitute tokens in the given string with placeholders, then put them back after transforming the string
- Pretty-bytes: Convert bytes to a human readable string
- Private: Utility for associating truly private state with any JavaScript object
- Process: Process information for Node.js and browsers
- Process-nextick-args: Always be able to pass arguments to process.nextTick, no matter the platform
- Progress-stream: Read the progress of a stream
- Promise: Consider the following synchronous JavaScript function to read a file and parse it as JSON
- Prr: A sensible alternative to Object.defineProperty()
- Pseudomap: A thing that is a lot like ES6 Map, but without iterators, for use in environments where for..of syntax and Map are not available
- Punycode: A robust punycode converter that fully complies to RFC 3492 and RFC 5891, and works on nearly all JavaScript platforms
- Q: A library for promises
- Qs: A query string parser that supports nesting and arrays. With a depth limit
- Query-string: Parse and stringify URL query strings
- Querystring: provides utilities for parsing and formatting URL query strings
- Querystring-es3: Node's querystring module for all engines
- Randomatic: Generate randomized strings of a specified length, fast. Only the length is necessary, but you can optionally generate patterns using any combination of numeric, alpha-numeric, alphabetical, special or custom characters
- Rc: Hardwired configuration loader
- Rcedit: Node module to edit resources of exe
- React: React is a JavaScript library for building user interfaces
- React-dom: React package for working with the DOM. Servers as the entry point of the DOM-related rendering paths
- React-dropdown: React dropdown component
- React-lazy-cache: A utility to lazily calculate and cache values in a react component based on props
- React-redux: Official React bindings for redux
- React-router: A complete routing library for react
- React-tooltip:
- Readable-stream: Streams3, a user-land copy of the stream library from Node.js

- Readdirp: Recursive version of fs.readdir with streaming api
- Read-pkg: Read a package.json file
- Read-pkg-up: Read the closest package.json file
- Redent: Strip redundant indentation and indent the string
- Reduce-css-calc: Reduce CSS calc() function to the maximum
- Reduce-function-call: Reduce function calls in a string, using a callback
- Redux: Predictable state container for JavaScript apps
- Redux-form: A higher order component decorator for forms using Redux and React
- Redux-thunk: Thunk middleware for Redux
- Regenerate: Generate JavaScript-compatible regular expressions based on a given set of Unicode symbols or code points
- Regenerator-runtime: Runtime for regenerator-compiled generator and async functions
- Regex-cache: Memorize the results of a call to the RegExp constructor, avoiding repetitious runtime compilation of the same string and options, resulting in surprising performance improvements
- Regexpu-core: source code transpiler that enables the use of ES6 Unicode regular expressions in JavaScript
- Regjsgen: Generate regular expressions from regjsparser's AST
- Regjsparser: Parsing the JavaScript's RegExp in JS
- Repeat-element: Create an array by repeating the given value n times
- Repeating: Repeat a string - fast
- Repeat-String: Repeat the given string n times
- Request: Simplified HTTP request client
- Resolve: Resolve like require.resolve() on behalf of files asynchronously and synchronously
- Retry: Abstraction for exponential and custom retry strategies for failed operations
- Right-align: Right-align the text in a string
- Rimraf: A deep deletion module for node
- Ripemd160: Compute ripemd160 of bytes or strings
- Run-series: Run an array of functions in series
- Rw: Now stdin and stdout are files
- Sax: An evented streaming XML parser in JavaScript
- Semver: The semantic version parser used by npm
- Set-immediate-shim: Simple setimmediate shim
- Sha.js: Streamable SHA hashes in pure JavaScript
- Shebang-regex: Regular expressions for matching a shebang line
- Sigmund: Quick and dirty signatures for Objects
- Signal-exit: When you want to fire an event no matter how a process exits
- Single-line-long: Module that keeps writing to the same line in the console
- Slash: Convert windows backslash paths into slash paths
- Sntp: SNTP v4 client for node. Connects to the NTP or SNTP server requested and returns the server time along with the round trip duration and clock offset
- Sort-keys: Useful to get a deterministically ordered object, as the order of keys can vary between engines
- Source-list-map: Fast line to line SourceMap generator
- Source-map: generates and consumes source maps
- Source-map-support: Fixes stack traces for files with source maps
- Spdx-correct: Correct invalid SPDX identifiers
- Spdx-exceptions: List of SPDX Standard license exceptions
- Spdx-expression-parse: Parse SPDX License expressions
- Spdx-license-ids: A list of SPDX license identifiers
- Speedometer: Simple speed measurement in JavaScript
- Split: Split a text stream into a line stream

- Sprintf-js: complete open source JS sprintfl implementation for the browser and node.js
- Sshpk: A library for finding and using SSH public keys
- Stream-browserify: The stream module from node core for browsers
- Stream-combiner: Turn a pipeline into a single stream
- Stream-consume: Consume a stream to ensure it keeps flowing
- Strict-uri-encode: A stricter URI encode adhering to RFC 3986
- String_decoder: The string_decoder module from Node core
- Stringstream: Encode and decode streams into a string streams
- String-width: Get the visual width of a string - the number of columns required to display it
- Strip-ansi: Strip ANSI escape codes
- Strip-bom: Strip UTF-8 byte order mark (BOM) from a string
- Strip-indent: Strip leading whitespace from each line in a string
- Strip-json-comments: Strip comments from JSON. Lets you use comments in your JSON files
- Style-loader: Style loader module for webpack
- Supports-color: Detect whether a terminal supports color
- Svgo: Nodejs-based tool for optimizing SVG vector graphics files
- Symbol-observable: Symbol.observe ponyfill
- Tabdown-kfatehi: An indentation-based tree markup language
- Tapable: Just a little module for plugins, for binding and applying
- Tar-stream: Streaming tar parser and generator and nothing else
- Temp: Temporary files and directories
- Throttleit: Throttle a function
- Through: Simplified stream construction
- Through2: A tiny wrapper around Node streams2 Transform to avoid explicit subclassing noise
- Timers-browserify: Timers module for browserify
- To-fast-properties: Force V8 to use fast properties for an object
- To-iso-string: Cross-browser toISOString support
- Touch: Like touch(1) in node, for all your node touching needs
- Tough-cookie: RFC6265 Cookies and Cookie Jar for node.js
- Traverse: Traverse and transform objects by visiting every node on a recursive walk
- Trim-newlines: Trim newlines from the start and/or end of a string
- Tty-browserify: The tty module from node core for browsers
- Tunnel-agent: HTTP proxy tunneling agent. Formerly part of mikeal/request. Now a standalone module
- Tweetnacl: Port of TweetNaCl cryptographic library to JavaScript
- Typedarray: TypedArray polyfill for old browsers
- Type-detect: Improved type of detection for node.js and the browser
- Ua-parser-js: Lightweight JavaScript-based user-agent string parser
- Uglify-js: JavaScript parser, mangler/compressor and beautifier toolkit
- Uglify-to-browserify: A transform to make UglifyJS work in browserify
- Underscore: JavaSCript's functional programming helper library
- Underscore.string: String manipulation extensions for Underscore.js javascript library
- Uniq: Removes duplicates from a sorted array in place
- Uniqid: Unique ID generator
- Uniqs: Tiny utility to create unions and de-duplicated lists
- Url: The core url packaged standalone for use with Browserify
- Url-loader: URLLoader module for webpack
- User-home: Get the path to the user home directory
- Util: Node.js util module

- Util-deprecate: In the web browser, a browser-specific implementation of the util.deprecate() function is used
- Validate-npm-package-license: Give me a string and i'll tell you if it's a valid npm package license string
- Vendors: List of vendor prefixes known to the web platform
- Verror: Richer JavaScript Errors
- Vm-browserify: VM Module for the browser
- Warning: A mirror of Facebook's Warning
- Watchpack: Wrapped library for directory and file watching
- Webpack: Packs CommonJs/AMD modules for the browser. Allows to split your codebase into multiple bundles, which can be loaded on demand
- Webpack-core: The core of webpack and enhanced-require
- What.extend: Sharpened version of node.extend as port of JQuery.extend that "Actually works"
- Whatwg-fetch: Promise-based mechanism for programmatically making web requests in the browser
- Which: Finds the first instance of a specified executable in the PATH environment variable
- Wide-align: A wide-character aware text alignment function for use on the console or with fixed width fonts
- Window-size: Reliable way to get the height and width of the terminal/console in a node.js environment
- Wordwrap: Wrap those words, show them at what columns to start and stop
- Wrappy: Callback wrapping utility
- Xml2js: Simple XML to JavaScript object converter
- Xmlbuilder: An XML Builder for Node.js
- Xmldom: A W3C standard XML DOM(Level 2 CORE)
- Xmlhttprequest: Emulate the browser XMLHttpRequest object
- Xtend: Basic utility library which allows you to extend an object by appending all of the properties from each object in a list
- Yallist: A doubly-linked list implementation
- Yargs: The modern, pirate-themed, successor to optimist
- Yauzl: Yet another unzip library for node
- Zip-stream: A streaming zip archive generator

Electron Libraries Used

- BrowserWindow: Create a window without chrome, or a transparent window in arbitrary shape
- ipcRenderer: send synchronous and asynchronous messages from the render process (web page) to the main process.

ReactJS Libraries Used

- ReactDOM: Serves as the entry point of the DOM-related rendering paths. It is intended to be paired with the isomorphic React, which will be shipped as react to npm
- React-router: Keeps your UI in sync with the URL (3 components)
    - hashHistory: uses URL hashes, along with a query key to keep track of state
    - Route: used to declaratively map routes to your application's component hierarchy
    - Router: Primary component of React Router. It keeps your UI and the URL in sync.

Redux Libraries Used

- appleMiddleware: suggested way to extend Redux with custom functionality
- combineReducers: Split reducing functions into separate functions, each managing independent parts of the state
- createStore: Holds the complete state tree of your app
- React-Redux: A generic library for connecting react components to Redux Store
- Provider: Makes the redux store available to the connect() component
- Redux-thunk (middleware): allows you to write action creators that return a function instead of an action.

Swift Libraries Used

- AppKit: Construct and manage your app's user interface for macOS.
- Cocoa: application development environments for OSX and iOS, respectively.
- Foundation: Access the essential classes that define basic object behavior, data types, collections, and operating-system services