**-**Logo%20Main%20200

**Boston University**

**Electrical & Computer Engineering**

**EC464 Capstone Senior Design Project**

User's Manual

Machine Learning Powered Electrical Scheduling

Submitted to

Alan Pisano

8 St. Mary’s Street

Boston, MA

(617) 353 - 6264

apisano@bu.edu

by

Team 28

EcoStrip Solutions

Team Members

Avishai Lean alean@bu.edu

Benjamin Axline baxline@bu.edu

Mateusz Górczak mgorczak@bu.edu

Konstantin Agrachev kostya@bu.edu

Submitted: 4/17/25

#### Machine Learning Powered Electrical Scheduling

#### Table of Contents

[Executive Summary 3](#_heading=h.5biteqd20rp4)

[1 Introduction 4](#_heading=h.n2y7rxqxy21z)

[2 System Overview and Installation 5](#_heading=h.grr4lnr0yxcn)

[2.1 Overview block diagram 5](#_heading=h.ksqjrnnc96q)

[2.2 User interface. 5](#_heading=h.ur1l8eqr633j)

[2.3 Physical description. 7](#_heading=h.fvr0if8xwzvt)

[2.4 Installation, setup, and support 7](#_heading=h.60h5i3s0hc0z)

[3 Operation of the Project 9](#_heading=h.4v1zfroz6qcw)

[3.1 Operating Mode 1: Normal Operation 9](#_heading=h.n2avj1t5cx7k)

[3.2 Operating Mode 2: Abnormal Operations 10](#_heading=h.jm0cb17ipt9b)

[3.3 Safety Issues 11](#_heading=h.w2d01wc12owd)

[4 Technical Background 12](#_heading=h.3jwnkq8xa2i)

[5 Relevant Engineering Standards 15](#_heading=h.cb9tijnhxdbc)

[6 Cost Breakdown 17](#_heading=h.i5mwsr6fowph)

[7 Appendices 19](#_heading=h.2dw4te1t1r52)

[7.1 Appendix A - Specifications 19](#_heading=h.pjcvhbkk8w9x)

[7.2 Appendix B – Team Information 20](#_heading=h.kpzqfyi1mjjz)

# 

# Executive Summary

Climate change represents an existential threat towards global health and populations. As a result, there has been an increased focus on electrification, as well as shifting more traditional electrical generation sources towards renewables like solar and wind. However, these new renewable or low-emission generation sources present an additional problem for grid systems as they are volatile and require optimal outside weather conditions to generate power. As these renewables constitute a larger percent of the total electrical generation, there are increasingly times in which renewables overproduce electrical demand and must be curtailed, while there are times in which demand peaks and renewables have little generation causing increased reliance on non-renewable sources.

Our final deliverable is machine learning algorithms that can accurately predict day-ahead renewable generation. These algorithms will be used to power a web application that allows a user to manage electrical consumption and schedule non-essential loads at times when there is maximum availability of renewable energy generation. Essentially, the application’s aim is to shift consumption from times of low renewable generation towards times with higher renewable generation. Finally, we will attach these algorithms to an outlet that can be controlled entirely by our web application as a proof of concept for these controls.

Our technical approach involves the creation, testing, and implementation of machine learning models and web applications. The energy generation sources models will be developed with various types of Neural Networks each written and tested independently. These models will be retrained on a regular basis with the latest data that is automatically pulled from Independent Systems Operators and Weather sources. Once created, these models will then be used to predict the following days based upon weather data and predicted total load from the ISO. The web application will be created using JavaScript/React for the frontend and Python/FastAPI for the backend. It will feature power consumption analytics, renewable predictions, and load control as its basic features.

Predicting renewable generation is an increasingly important field of research and one in which our team has drawn inspiration towards our algorithms. The innovative aspect of our project is to change electrical loads towards times of renewable generation.

# Introduction

Machine learning (ML) is defined as “a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behavior”.In the context of our project, we will use ML techniques to find patterns in the energy generation source outputs and use these patterns to predict the future generations.

Electrical Grid Systems are the way in which electrical power reaches the consumer, this is described in Fig.1.

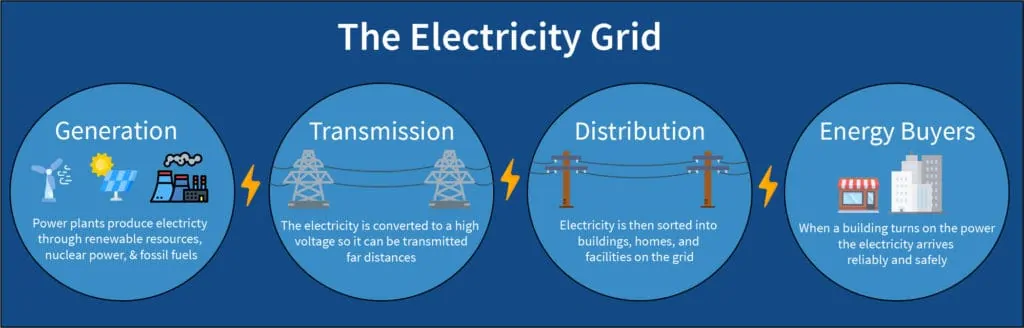


Fig. 1 : Electrical Grid Systems

Independent system operators are the group that oversees the process described above and are instrumental in regulating energy generation to match supply and demand. This process is described in Fig 2. New England’s ISO will be our main data source.

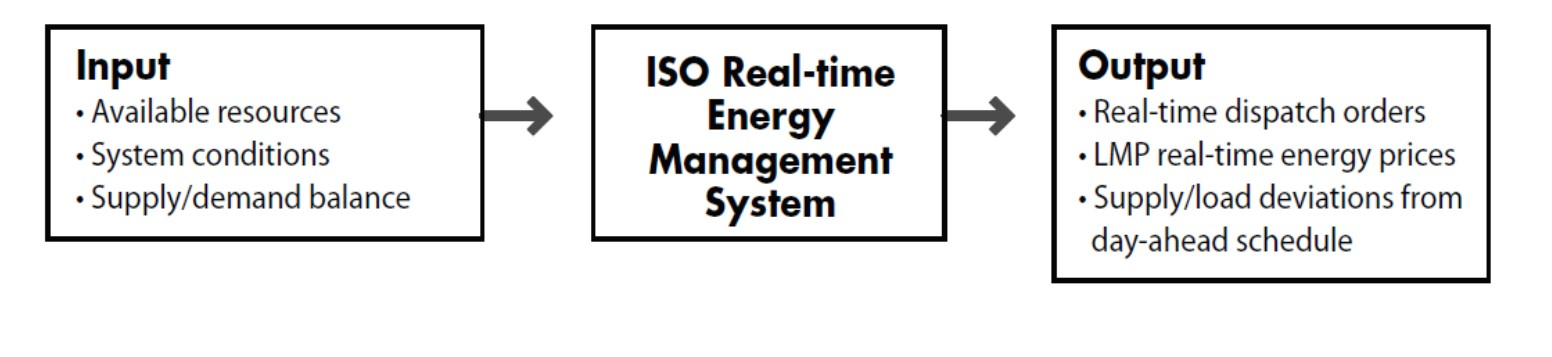


Fig. 2 : Independent System Operators

Generally, our project will be achieved through the combination of our Machine Learning Algorithms and Web application. First, we create individual ML algorithms for each generation type based upon the latest weather data and power data. We then combine each algorithm and predict the next day of renewable energy generation. Once we have created these predictions, we upload them to github where the WebApp can then pull this data to display the predictions. Finally, based on these predictions as well as power consumption analytics, the application can provide recommendations for the best times to consume electrical power.

Our project allows the user to easily schedule electrical consumption to occur at times in which there is a surplus of predicted renewable energy production. This will allow users to track their power consumption, and we will report to them the likely carbon savings as a result of switching to our EcoStrip Solutions. This innovative interconnection between the ML backend, web application and physical devices such as the Kasa Smart Plug serves as the basis of our project.

# System Overview and Installation

## Overview block diagram

In our block diagram it shows the flow of information from data intake to outputs to strip outlets. First our backend calls for data from the ISONE as well as Visual Studio Weather Data and then combines to create our training set. This training set is used to create the model that then is outputted to the github which the web uses to display forecasts. The user interacts with the web model and can then interact with the various devices.

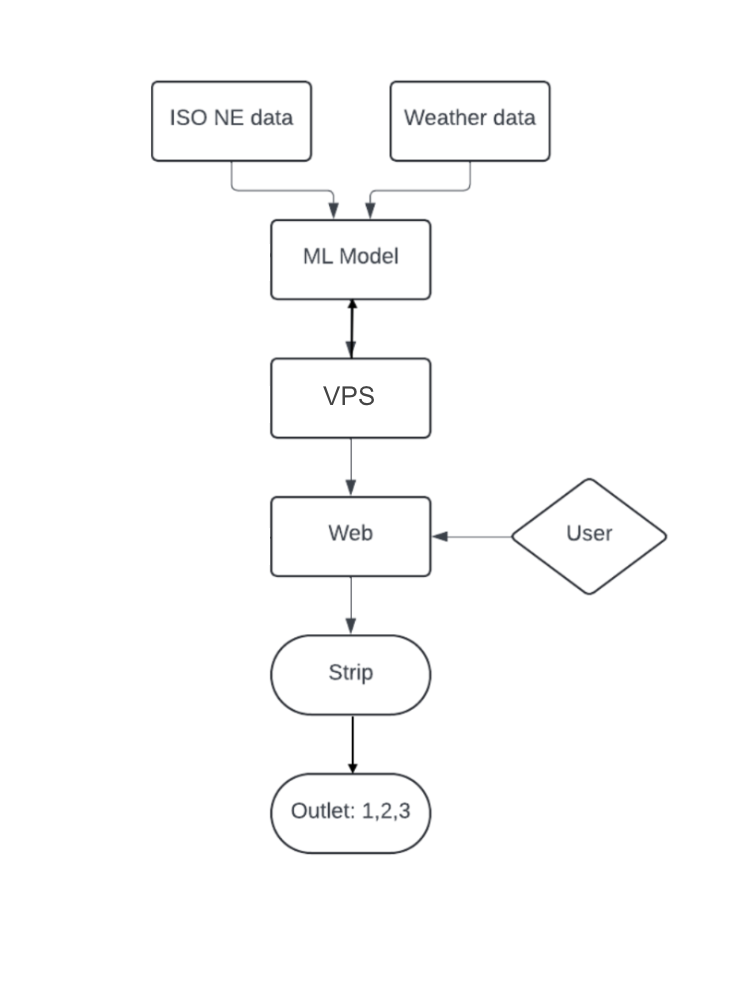


Figure 3: System Block Diagram - New One

## User interface.

Our user interface is displayed on the website ecostripsolutions.com. It revolves around the display of various graphs related to the forecasting of renewable energy generations. When users first visit the website they are greeted with predictions as well as the panel that allows the user to add devices and view notifications regarding their energy use. These graphs include generation predictions for the next 48 hours, generation breakdown by source of last year, as well as the generation prediction (percent renewables verses non renewables) for the next forty-eight hours. See screenshot below for depictions of the graphs. Notably, the user cannot interact with these graphs, they are just for absorbing the information.

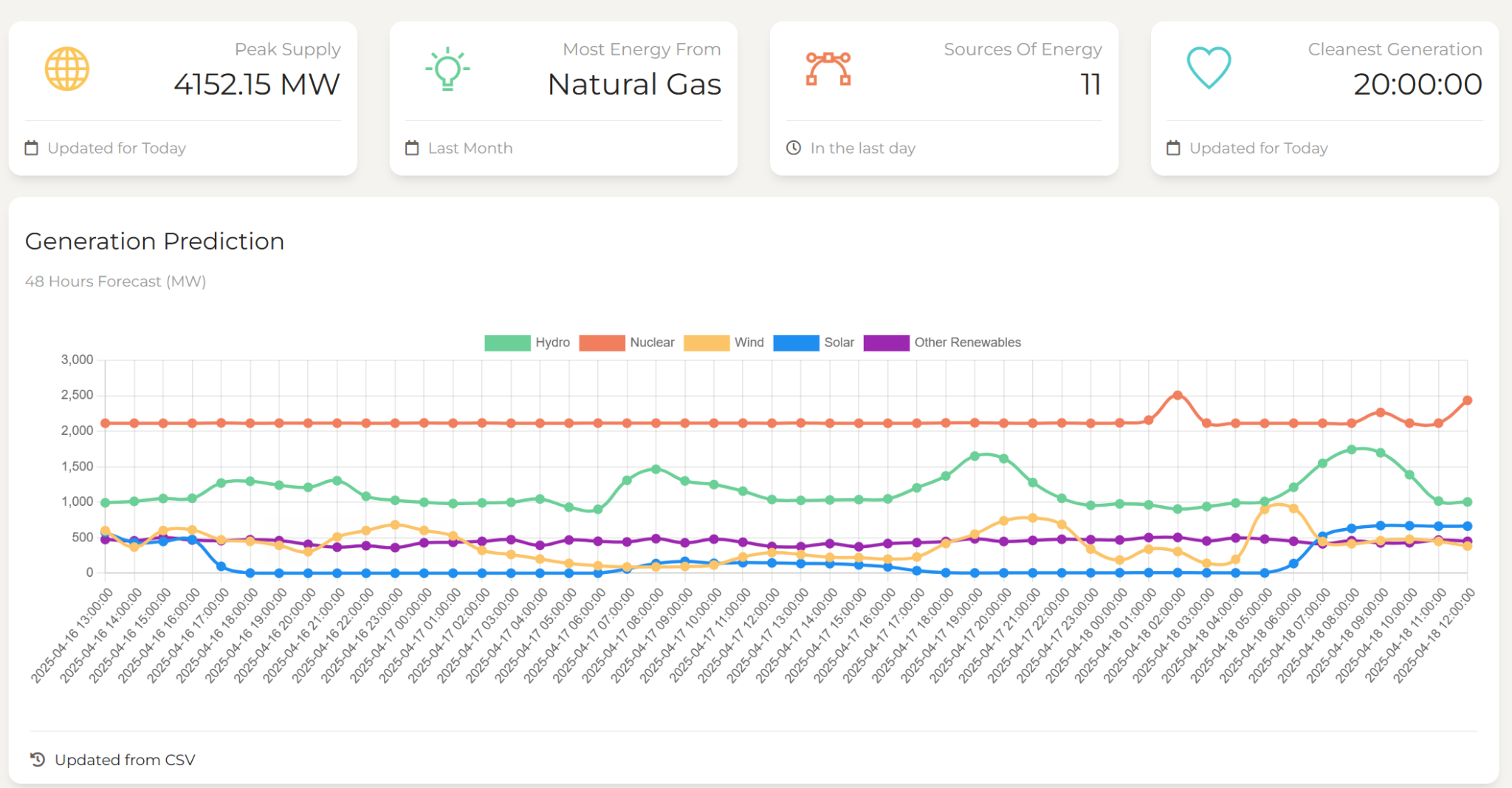


Figure 4 : Website Landing Page

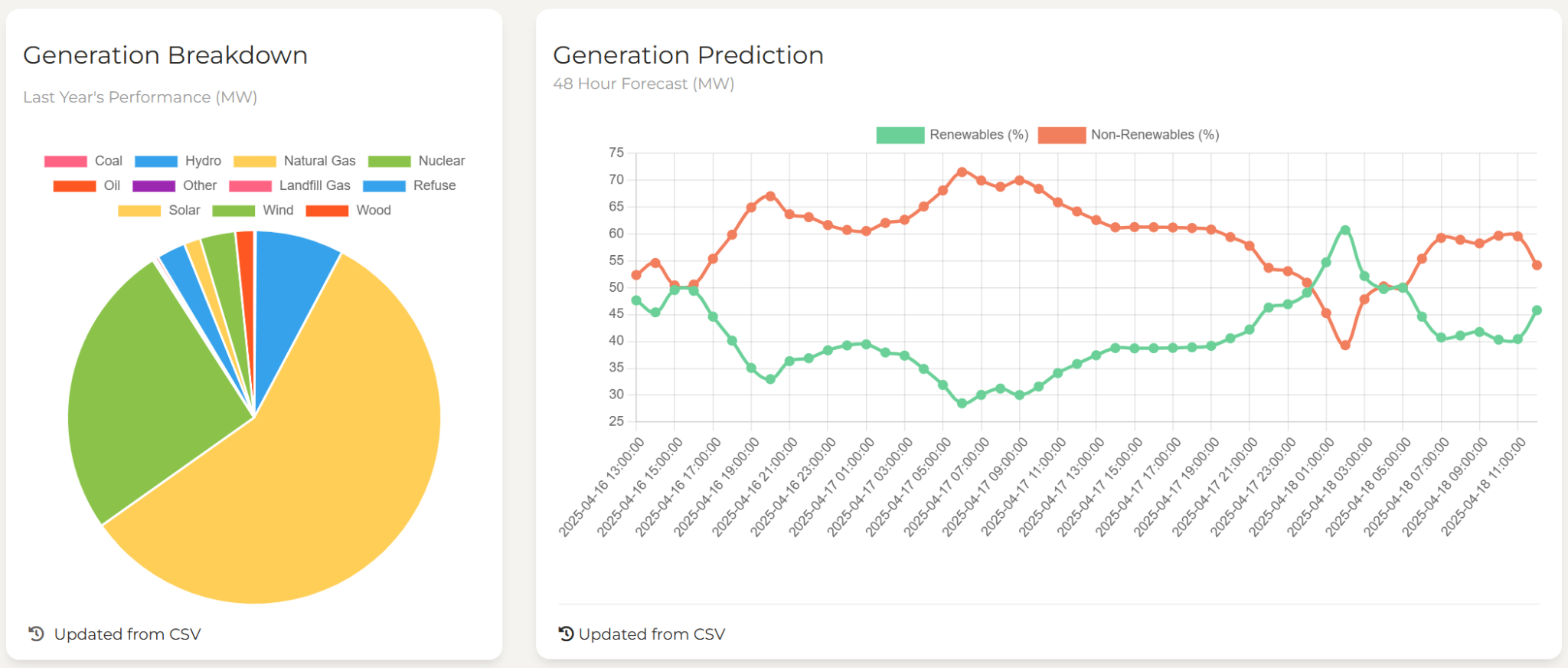


Figure 5: Prediction & Breakdown

In order to interface with the website the user must add a device using our enter device address feature. This requires the device to have an IP address, API key, or wifi connections. Once this connection has been made the device will be saved and the user can interface with the device scheduling feature that allows the user to choose times to have their device on/off with recommendations generated from the best times.

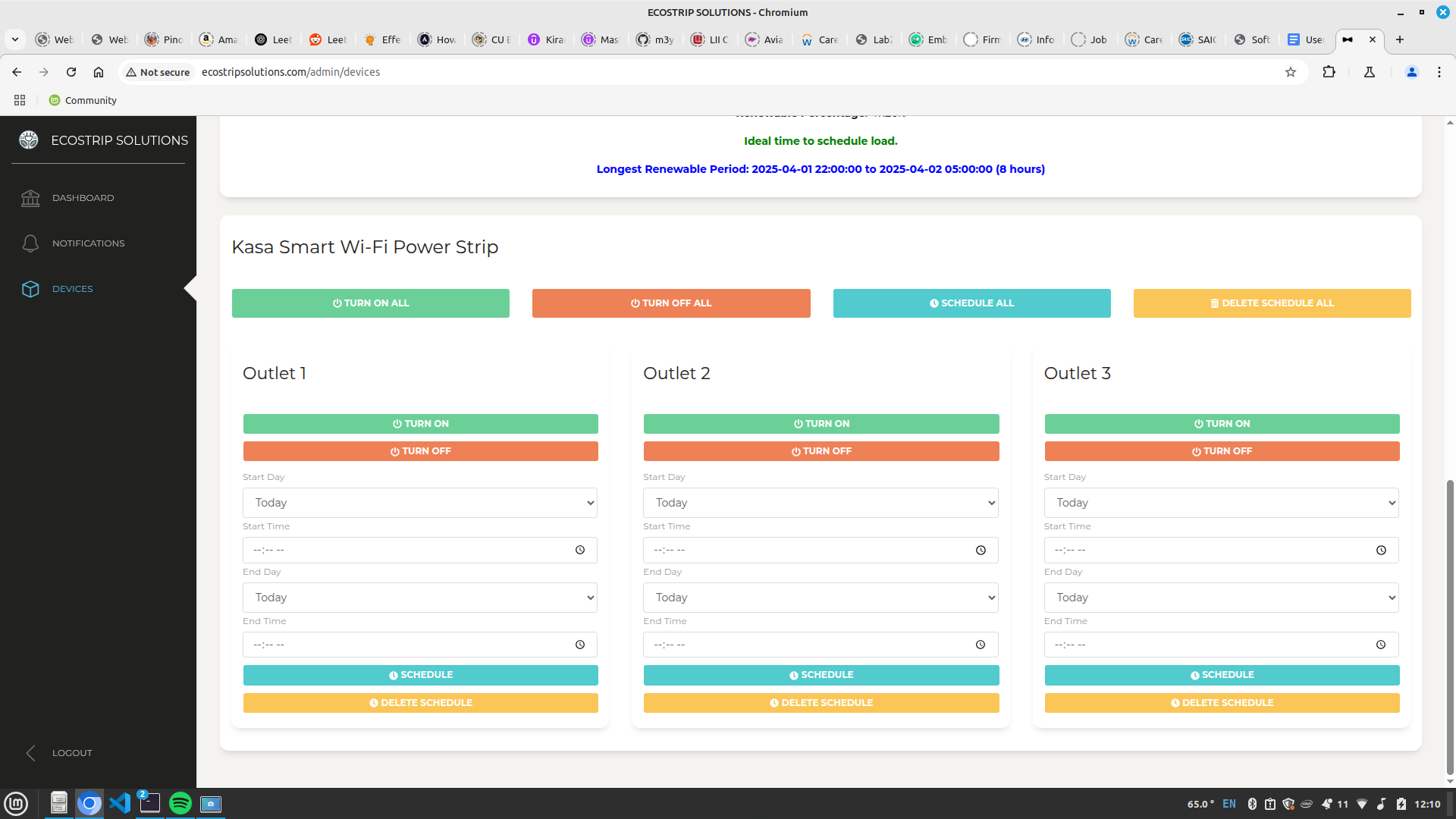


Figure 6: Scheduling Feature

## Physical description.

Our project is almost entirely a software project with our physical components being our server (desktop pc in the library) and our physical controlled device.

The only device (Kasa Smart Outlet) is controlled through the IOT though nGrok.



Figure 7: Kasa Smart Strip

## Installation, setup, and support

In order to use our website the user does not have a large installation or setup process. Simply visiting the website creating an account using SSO log-in procedures is easy. Behind the scene there is a more complicated installation setup made from many different Python libraries as well as JavaScript and React.

Setup for the Kasa Smart Strip:

1. Plug the powerstrip into an outlet.
2. Download and install the Kasa app onto your smart device.
3. Open the app and follow the steps to connect your powerstrip to your router.
4. Once connected, go to your router dashboard and retrieve the IP address of your powerstrip. Note this down for later.
5. Create an account at <http://ecostripsolutions.com>.
6. Create an account on ngrok and install it on your computer: <https://ngrok.com/downloads/>.
7. Go to <https://dashboard.ngrok.com/domains> and claim a static domain. This will serve as the address through which your powerstrip can be accessed via the website.
8. Run ngrok with this command, where sample\_url is the name of the static domain you claimed:

ngrok http http://localhost:8080 --url="sample\_url"

1. Run the provided FastAPI script with this command, where DEVICE\_IP is the IP address you noted down earlier:

DEVICE\_IP=192.168.xxx.xxx uvicorn app:app --port 8080 --reload

1. Now your powerstrip is ready to be used with the website. Go on the website, login, and go to the Devices tab. Where it says “Enter Device Address”, enter the static ngrok domain. This gets saved for user convenience.
2. Assuming no connectivity issues, new tabs should appear, tabs through which you can see real-time energy data and control the powerstrip.

# Operation of the Project

## Operating Mode 1: Normal Operation

In normal (device not connected) operation the user can access our website and use certain functionalities within the website. Optimal operation for a user would likely follow the following procedure:

1. Visit ecostripsolutions.com using a preferred web browsing application.
2. Navigate to the Login Button on the left side of the screen as shown below.

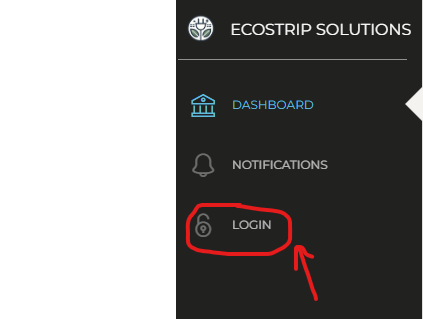


Figure 8: Navigate to Login Page

1. Proceed to either login using an existing user or register for a new account using the register here button. Shown below:

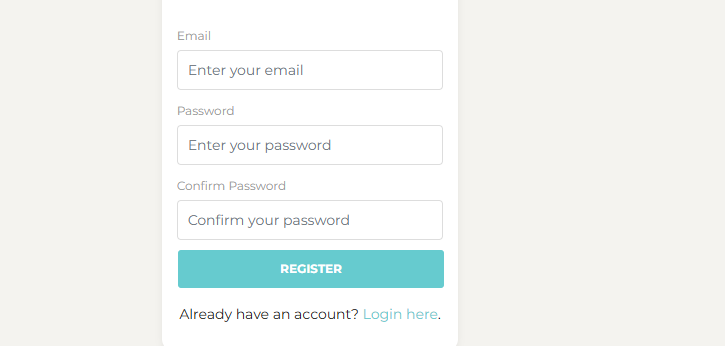


Figure 9: Login Feature

1. After creating or logging in to the web application, the user can then interface with the website to add a device if desired (see Operation Mode 2).
2. If the user does not want to use a device they can still observe the generation predictions, peak renewable supplies, largest energy source, cleanest generation time, and 48 hour generation prediction divided by renewables and non-renewables.
3. From these graphs and information the user could then independently choose how/when to plan their day of electrical planning and scheduling.

## Operating Mode 2: Abnormal Operations

Operating Mode 2: Device Connected

Following step 4 from Section 3.1, the user now has access to additional functionalities to control the powerstrip. In this operation mode it is assumed that an existing device has already been set up. Please refer to section 2.4 for best practices regarding establishing device connection between website and physical device. There may be situations where the previously connected device might disconnect due to internet connection issues, which is why right after entering the device address, the website pings the device to check if it’s connected before displaying anything else. If the powerstrip is disconnected, an error message is displayed. Following the successful establishment of connection the following best practices can be followed to maximize the use of the device.

1. After successful integration of a given device, check the current renewables generation output. Shown below this pie chart represents the current (within fifteen minutes) realtime generation split in the NEISO. This can be used to quickly activate power if the percentage is high enough.

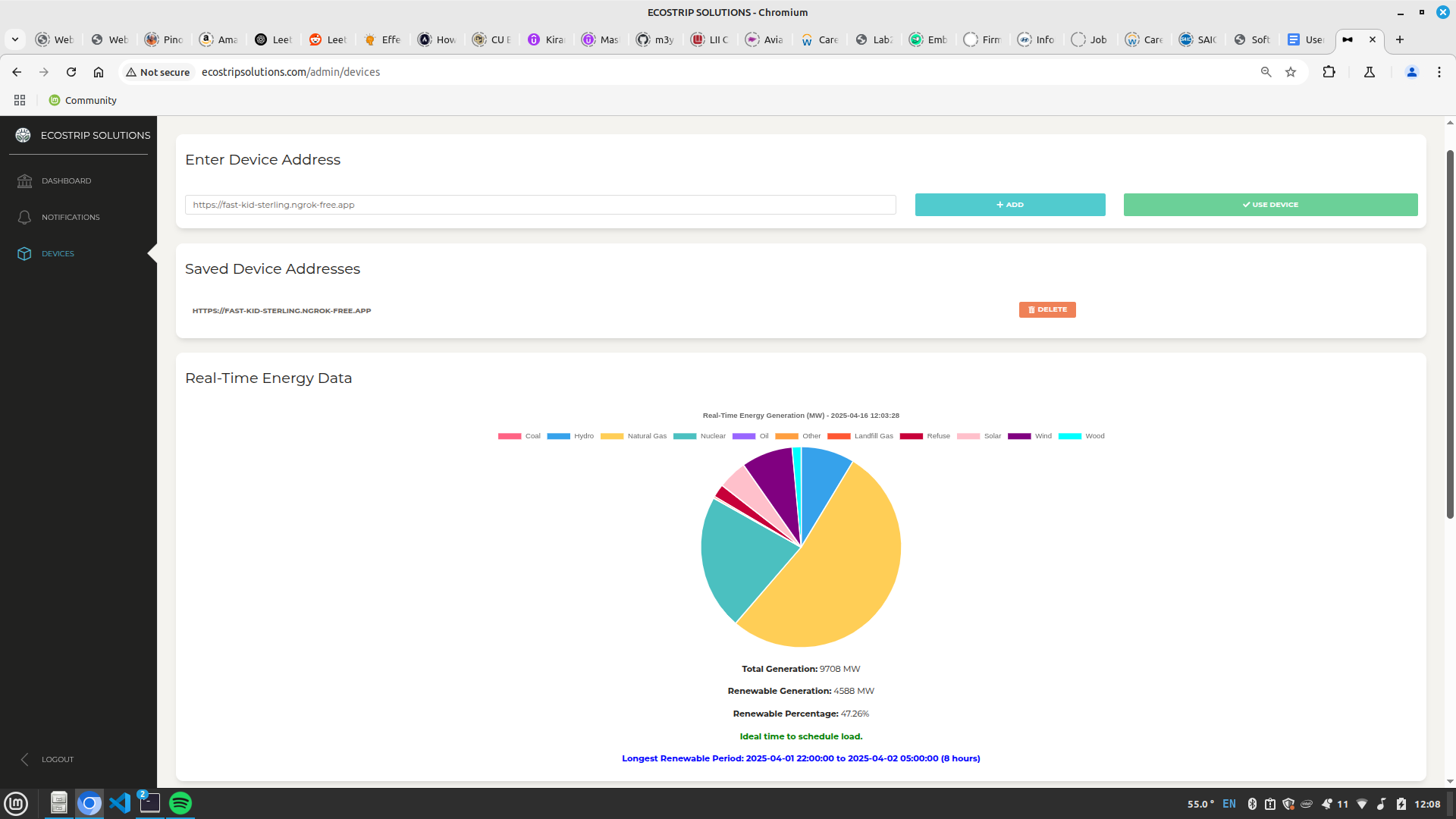


Figure 10 : Real Time Renewables

1. The website will also offer the user the largest block of time with significant (>50%) renewable generation in a pop up.
2. With this information and the information in the landing page the user is now educated to make decisions and can execute based on the scheduling figure below.

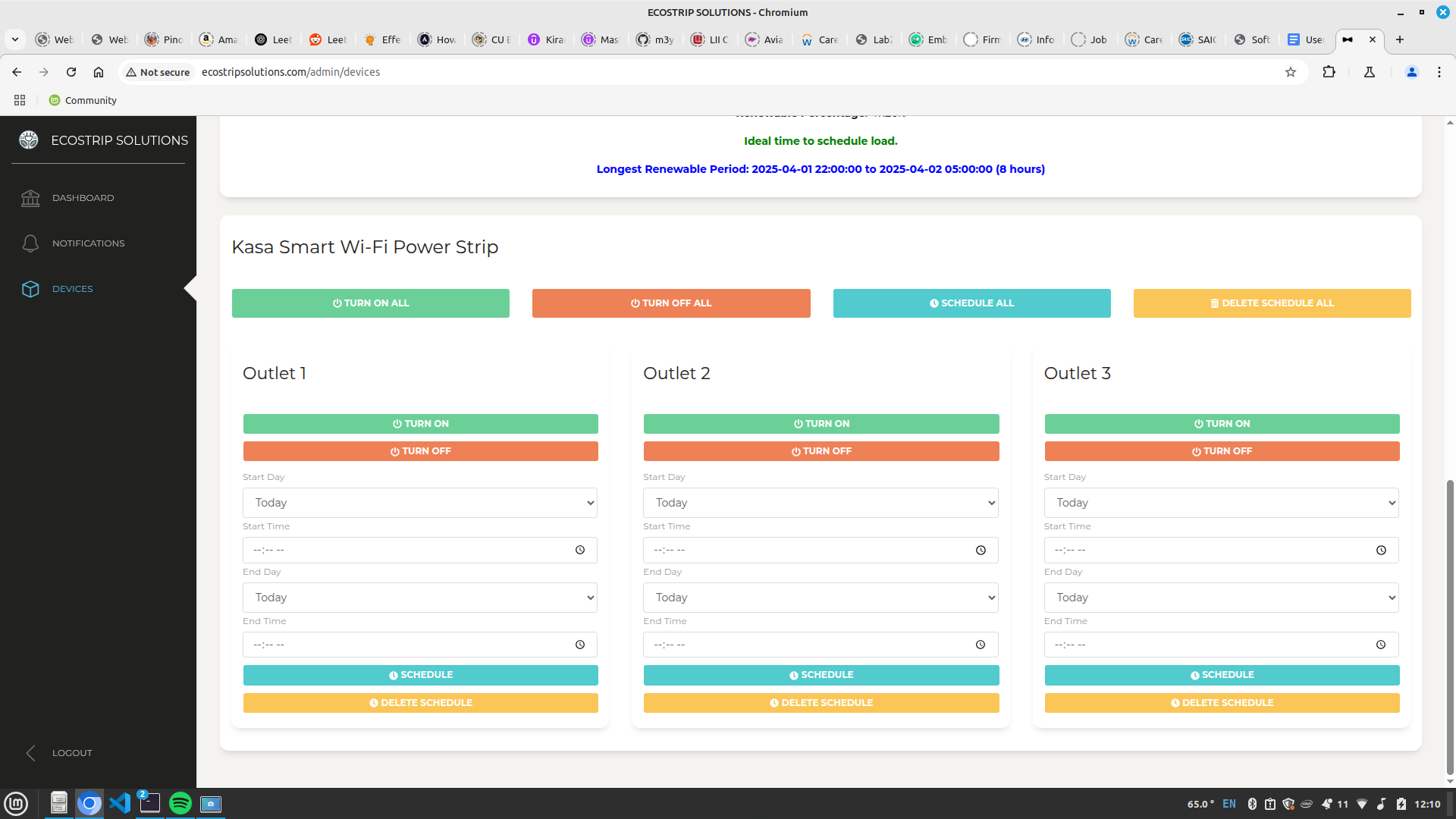


Figure 11: Scheduling Feature

## Safety Issues

Generally, since our project is software-oriented, there are no great physical harms that can affect people. However, two effects could potentially exist.

First, since we allow for a login, we have personal client data for our users. This could be considered a cybersecurity risk, as this data needs to be encrypted and protected. To combat this, we have added the SQLite Encryption Extension (SEE) plug directly into SQLite’s pager layer so that every page read from or written to disk is automatically decrypted or encrypted using your key. When you compile SQLite with SEE enabled, the pager calls a small “codec” hook on each 4096‑byte page.

Second, with the physical device control, we only have two modes - on or off. Issues should not arise from this (assuming a fully working powerstrip), however, users should still take care in deciding what is safe to use with the powerstrip (e.g. fridges should not be plugged into powerstrips, generally speaking).

# Technical Background

Our machine learning pipelines are powered by data received by the ISO New England – the organization in charge of managing and controlling power distribution – combined with detailed historical and 15-day forecasted weather data from Visual Crossing. The data is then preprocessed, compiled, and merged into a combined CSV which serves as the training input for our models.

The machine learning backend includes separate models for each renewable and clean energy resource including hydroelectric, landfill, nuclear, refuse-derived fuel, wood, solar, and wind. Each model is built using tensorflow feedforward neural networks that are tailored to the needs of that model and the characteristics of the energy source. At the core of each model, we utilized time-series forecasting principles to predict the energy output of each energy source. Our engineered features such as energy sum, and lagged energy (1-day, 2-day) that allow the model to interpret the desired energy source and infer patterns without directly taking in the label. They also help the models capture temporal dependencies and trends over time. Once the features have been implemented the model is then tailored to the individual source before creation based on the needs and complexity of the energy source. Each model has a customized approach to achieve the best results. For example, our solar generation model includes additional filters to separate the data by season and then implement a deeper and more complex feedforward model. This model is shown below (Figure 12). It involves a learning rate scheduler, early stopping, BatchNormalization, kernel regularizers, LeakyReLU, and dropout layers. All of these techniques were chosen based on performance and involved selecting the hyperparameters that provided the best performance. Through hyperparameter tuning, we were able to achieve the best results.

In contrast, our nuclear energy generation model is much simpler due to its predictable and stable characteristics. As a result, its model architecture is much simpler, only relying on basic activations and dense layers. Historically, nuclear energy generation is generally steady under normal conditions, making it much easier to predict that other sources that aren’t as steady. This can be seen in figure 13.

Once each model is trained, we save it in a .h5 format. We then have a general script that then combines these individual models into predictions that can be combined into a CSV file. This CSV file is then pushed to the cloud.

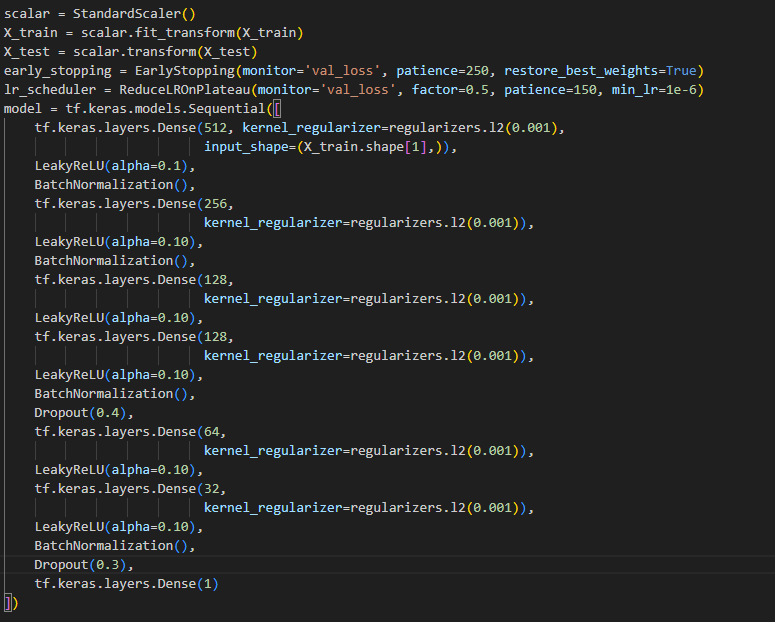


Figure 12: Solar Machine Learning Model

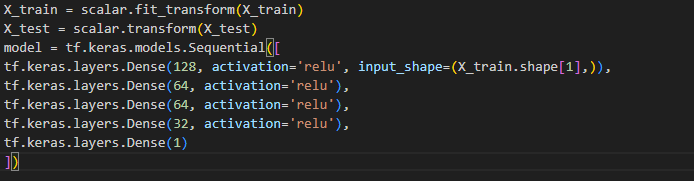


Figure 13: Nuclear Model

We built the front end on Creative Tim’s Paper Dashboard React template to take advantage of its responsive layout and built‑in chart components, then tailored it to our needs by importing our 48‑hour generation forecast CSV directly into dashboard.js and parsing it into React state so we could feed it straight into the template’s <Line> and <Pie> components for smooth area‑ and pie‑chart visualizations. We also refined the UI for clarity and accessibility by adjusting font sizes, contrasts, and we positioned the most critical charts “above the fold” to ensure immediate visibility. To support live updates, a React effect periodically pulls new predictions from our github main branch pushes fresh data into the same chart components, so everything stays up to date. For deployment, we provisioned a Hostinger VPS and configured NGINX as our reverse proxy and static‑file server, serving the optimized React build, handling HTTPS, and proxying API calls to our back end for performance and reliability. Finally, we used FastAPI on the server side to connect the web application to the TP‑Link power strip, enabling users to control outlet power directly through the dashboard.

In our frontend website design, we also incorporated a devices panel, where users can add their powerstrips, toggle power to them, and schedule loads for them. Usage of this devices panel is described in better detail in the installation/setup section.

# Relevant Engineering Standards

As a software-driven platform integrating machine learning, real-time analytics, and smart outlet control, EcoStrip Solutions adheres to a range of established engineering standards. These standards were deliberately selected to ensure the reliability, scalability, and usability of the system across different modules, while maintaining security and ensuring interoperability with external systems and services.

ISO 8601 – Standardized Date and Time Representation

A critical aspect of our platform involves the accurate tracking, logging, and synchronization of time-based data from multiple sources, including smart outlets, user commands, and model predictions. We standardize all timestamps using the ISO 8601 format (YYYY-MM-DDTHH:MM:SSZ), which ensures unambiguous interpretation of date-time values across our backend and APIs.

Our backend processing pipeline stores all temporal data in Coordinated Universal Time (UTC) to maintain consistency, especially when interfacing with external APIs, scheduling tasks, and aggregating metrics for machine learning inference. However, for improved user experience, particularly for users located on the East Coast of the United States, we convert these UTC timestamps to Eastern Time (EST/EDT) for display in dashboards, reports, and notification systems. This required the implementation of robust time-handling logic capable of dynamically adjusting for Daylight Saving Time and mitigating common errors such as timestamp drift and timezone mismatches.

JPEG – Image and Video Format Standard

Where applicable, such as in visual diagnostics or system snapshots, our platform uses the JPEG (Joint Photographic Experts Group) standard for image compression. This widely adopted format enables high compression ratios without significant quality loss, allowing us to efficiently store and transmit images (e.g., power strip status photos or thermal images from external integrations) while minimizing bandwidth and storage costs.

RESTful APIs – Communication and Modularity

Our system architecture is built on REST (Representational State Transfer) principles, enabling clean, stateless communication between system components. The machine learning models, device control modules, and frontend interfaces all interact via RESTful endpoints, ensuring modularity and scalability. This structure facilitates third-party integrations, parallel development, and easier maintenance. Data is transmitted using standard HTTP methods with JSON payloads, ensuring compatibility across a broad range of platforms and languages.

Single Sign-On (SSO) – Streamlined and Secure User Authentication

Security and usability are paramount in managing user access, particularly given that users can remotely control smart devices. We employ Single Sign-On (SSO) protocols to allow secure and seamless authentication across all parts of the platform. This eliminates the need for users to manage multiple sets of credentials, reduces authentication errors, and improves onboarding efficiency. Our SSO implementation supports industry-standard protocols such as OAuth2 and OpenID Connect, enabling compatibility with institutional logins and enterprise identity providers.

Secure Login Infrastructure and Session Management

Beyond SSO, we implement robust security practices for all user authentication flows. This includes:

Password hashing with salting to prevent credential leakage in the event of a breach,

Rate-limiting and IP throttling to guard against brute-force attacks,

HTTPS encryption for all data in transit,

Session timeouts and token expiration to prevent unauthorized reuse of session credentials.

These practices align with cybersecurity best practices and regulatory standards, ensuring that both personal data and system control pathways remain secure.

# Cost Breakdown

| Project Costs for Production of Beta Version (Next Unit after Prototype) | | | | |
| --- | --- | --- | --- | --- |
| Item | Quantity | Description | Unit Cost | Extended Cost |
| Kasa Smart Outlet | 1 | The physical smart outlet is bought from amazon and costs 26.00 | 26.00 | 26.00 |
| Server Costs | Data | Costs to retrieve data from visual studio code and from NEISO | 0.25 | 0.25 |
| TP-Link AC1350 Router | 1 | Router used to bypass connectivity complexities with university internet | 20.00 | 20.00 |
| Month of VPS hosting | 1 | One month subscription to host our virtual private server | 18.00 | 18.00 |
| Year of Domain ownership | 1 | One year subscription to own our website domain | 9.00 | 9.00 |
| Beta Version-Total Cost | | | | 73.25 |

Each product requires the machine learning backend to be trained daily, which involves pulling and updating data from their respective sources. This would involve a small charge of around 25 cents. Theoretically, if multiple users were in the same region as each other, the models could be trained once for all users meaning that this 25 cent charge would only happen once per day. Each user would need the smart outlet and router for the system to be fully functional.

We also need to pay to host our webapp on a virtual private server. We decided to go with the popular hosting service called Hostinger. This is because of their extensive instructions on how to correctly and successfully host a webapp. Hostinger also allows to purchase a domain, therefore it simplified the process of hosting our website to only one provider.

# Appendices

## Appendix A - Specifications

Give these as a list or table. These should quantify the performance that the project

Final Deliverable Specifications

Machine Learning Model Accuracies - As of 4/16/25

| Generation Source | Test Loss (MW) | Percent Error (%) |
| --- | --- | --- |
| Hydroelectric Power | 28.488 | 2.37 |
| Landfill | .636 | 2.48 |
| Nuclear | 6.39 | 0.191 |
| Refuse | 4.175 | 1.64 |
| Solar | 9.97 | 5.41 |
| Wind | 19.30 | 3.37 |
| Wood | 3.51 | 1.76 |
|  |  |  |
| Total | 72.50 | 2.02 |

Front End Displays - Checkboxes

| Function | Status |
| --- | --- |
| Log In System | In Loving Memory of Square Checkbox @ tonsky.me |
| Power Forecasting Display | In Loving Memory of Square Checkbox @ tonsky.me |
| Scheduling Works | In Loving Memory of Square Checkbox @ tonsky.me |

## Appendix B – Team Information

**Team Information**

**Avishai Lean – Electrical Engineering**

**Project Management & Machine Learning Engineer**

**1898 & Co.**

**Mateusz Gorczak – Electrical Engineering**

**Lead Front-End Developer**

**Eversource**

**Konstantin Agrachev – Computer Engineering**

**Lead IOT Engineer**

**U.S. Air Force Software Directorate**

**Benjamin Axline – Computer Engineering**

**Lead Backend Engineer**

**Boris FX**