Power.Log Technical Document

R Costa-Tré, J Jandrell, A Kapp, K Ngwenya

10 June 2022

Abstract

Power.Log is a website (which may be converted into a web app in future) that allows users to log their average power/energy consumption and compare it to that of other users in other vehicles. This document outlines the high-level design of the product envisaged and the technical architecture and operation of the first prototype. The current prototype is then evaluated against the final vision to suggest the direction of future development and design.

1 High-level design

1.1 Overview

Power.log is a website/web application which allows users to record the energy/fuel they use on transport and compare it to the energy they would have used travelling the same distance in different vehicles. All data used in this comparison will be from a database of user logs to ensure a fair comparison. The site will make the users aware of their energy usage and give them a sense of how well they are doing to save energy compared to the general public.
Once the general log system is in place Power.Log will be expanded to track the average travel speed of users by using smartphone geolocation. Information such as travel will put fuel consumption into context. After building up a sufficient database (provided by users) the energy comparison system will be able to predict the energy the user would have spent in other vehicle types based on travel speed as well as distance. This should provide a more valuable comparison if the user is considering changing vehicles however it will not replace the purely distance-based comparison. Avenge speed data could also be used to suggest how much energy a user would save if they were able to avoid rush hour.

1.2 Purpose

The purpose of this project is to help users track the amount of energy they burn commuting on a day-to-day basis using their vehicle and compare this to the estimated energy usage of others' vehicles. This allows the user to understand how much energy they consume on transport while putting it into context. Comparison with others allows users to gauge the energy use of their vehicle based on the potential performance of alternatives.

1.3 Intended Audience

This system is aimed at users who wish to measure and decrease the size of their carbon footprint or reduce spending in the context of transport and their daily commute. These users are considered to be motorists with either environmental or fantail concerns. Private vehicle owners are targeted because they have the largest individual impact on general energy consumption.

1.4 Intended Use

This system is intended to allow users to quantify the amount of energy they use in the form of fossil fuels by converting the amount of fuel used or purchased into units of energy. Users are expected to log on to the website and enter travel information (distance travel and energy consumed). Users may also use the site to view their energy usage relative to others.

1.5 System Features and Requirements

1.5.1 Functional Requirements

- A graphic user interface (GUI) which allows the user to easily register, log in, and submit fuel logs access their power usage and compare this to alternative means of transport.
- Functions with the ability to calculate how much fuel is consumed based on distance and refuelling information.
- Function which converts the fuel in litres into energy in Kilowatt-hours.
- Data storage and data access: possibly local but ideally server-based

1.5.2 Non-functional Requirements

The display interface needs to be clear and display data in a way that is understandable to the user, however greater priority is given to calculations and higher accuracy for the data. The graphic user interface (GUI) should allow the user to select and view their energy consumption in units they are familiar with (kWh) and compare it to that of other vehicles.

1.6 Design Considerations

1.6.1 Local Application vs Website

When developing a system such as Power.Log there are two possible platforms which may be used to support the product: websites/web applications and local applications.

1.6.2 Local Applications

A local application would be made for mobile as the user is more likely to have a smartphone at hand while driving. This would enable the use of location services which would allow for simple location tracking and possible automation of the fuel logging process. Drawbacks to this are the complicated process of developing for multiple platforms and the limitations that come with storing files on the user's local device.

1.6.3 Web Applications

A web application would cater for users who prefer to work on their desktops as well as those who prefer their smartphones. This comes with the advantage of not having to develop for multiple platforms as web applications are automatically cross-platform. A web service would use a collective database for all user data which is beneficial when considering the concept of shared energy usage comparison. This would require a web connection which may be inconvenient, however, it would allow for social features such as comparing fuel consumption with other users. Unlike a native application, a website/web application is low commitment (does not require a sure download). A website also greatly reduces development complications and thus costs because the application would not need to be developed for multiple platforms. These benefits led to the team choosing to move forward with a web application instead of a mobile application.

1.6.4 Data Storage Location

User data could be sorted locally on the user's personal browser or a server. In browser storage is easier to develop and removes the need for a constant Internet connection, however, it complicates comparative data collection. A sever ensures that all user data is in a single location making it easy to fetch and compare on request. Thus, the current model uses a separate database however future work may investigate the possibility of a hybrid system.

1.7 Private or Public Transport

When tracing the energy used for travel there is an option to trace energy used by private or public transport (or to track both). Simply tracking both private and public transport is not trivial because the energy usage of a public vehicle is amortised (shared between multiple users) while the energy of an individual vehicle is only for the driver. Thus, private transport is chosen because following the logic above, private drivers are fully responsible for their energy usage and thus have a greater individual impact.

1.8 Programming languages

Traditional languages are used for the front end because they are well-documented standards. Alternatives, such as hosting a python front-end with flask are comparatively inconvenient and provide no apparent benefit.
Python is used in the back end because it is commonly used for data analysis and manipulation.

1.9 Potential Risks

- **Delay Due to Other Commitments.** The development teams are all full-time students and some work part-time jobs. There is a risk of cascading delays if tasks tend to more time than initially budgeted for. Good agile project management is vital in minimizing this risk.
- **Irregular Fuel Logging.** The aim to improve the consumption calculation algorithm based on user fuel logs requires the user to be diligent in their fuel logging. A low logging frequency will lead to discrepancies and impossible results. Rewards to incentivise fuel logging may need to be considered.
- **Inaccurate Fuel Logging.** The fuel logging process requires user input. This data will be used to improve predictions. Inaccurate input will lead to the development of inaccurate petitions. Some form of statistical analysis will need to be conducted to log records to confirm that they are within realistic estimates and not subject to the user making a human error when inputting values or simply forgetting to update the logger.
- **A Lack of Participation** The system relies heavily on user data. Without a community of dedicated users, the system is virtually useless. For this reason, Power.Log will offer no information to a user who has not already logged some of their own data. Power.Log will also include social media sharing and integration to improve brand awareness and encourage users to get their friends involved.

1.9.1 Challenges

- **Inexperienced Development Team.** The group has never produced a web application. There will likely be a steep learning curve and several errors which more experienced developers would know how to avoid. This increases the time required to produce a quality product.
- **Accurate Energy Estimates.** countless variables impact the fuel/energy consumption of a motor vehicle. Managing these variables and using them to produce reliable estimates will require a great amount of care. Precise calculations are virtually unattainable. Verifying results with a log should provide data that can be used for improvement.
- **Numerous Vehicle Models.** Each vehicle model from each manufacturer has a unique set of variables that affect energy consumption. These could be anything from fuel type and engine size to drag. Collecting and managing a comprehensive database for all models is an ongoing challenge. Especially with new models continually being released

2 First Prototype

front-end consists of HTML, CSS, and JavaScript with charts.js supporting data visualization. The back end runs on Python. The blackened and front-end and integrated using Flask and Flask-Cors.

This prototype is a rough proof of concept and is intended to demonstrate the basic structure and features. of the application.

2.1 Running the First Prototype

Power.Log is yet to be deployed to a public site but it can be run from the git-hub repo

2.1.1 Prerequisites

- Visual Studio Code with the 'live server' instalment or an equivalent environment capable of hosting a local HTML site.
- Python3 and the python venv module
- Use a Linux-based operating system
- Any browser other than internet explorer

2.1.2 Setup

- Activate the .venv virtual environment at the root of the repository by typing '.venv/bin/activate' while in the root directory (ELEN_PROJECT).
- Navigate to the /FINAL/BACKEND directory
- in FINAL/BACKEN run app.y from the command line using 'python3 app.py'
- In a separate terminal session, navigate to /FINAL/WEB _CODE/HTML and open index.html in Visual Studio Code
- Then right click and select the 'live server' option to run the site in your browser.

2.2 Technologies and Frameworks

2.2.1 Flask

Flask is a web framework written in Python. It features a collection of useful tools that make creating web applications with Python easier.

2.3 Charts.js

Chart.js is a free JavaScript library for data visualisation. It will allow Power.Log to use graphs and charts to show users how their fuel consumption compares with that of other users.

3 System structure

3.1 Systems

The back-end is made up of many different Python scripts, each making up a part of the system. These systems are the Energy Calculator, Login System, Unit Converter, Database, Encryption System, and Data Fetcher.

3.2 Energy Calculator

The energy calculator system works with two scripts. energyCalc.py and unitConverter.py. With these scripts, it can use the data gathered from fuel logs and return the user's total fuel consumption between their previous and most recent logs. This information is also converted from Litres to Kilowatt Hours for the user.

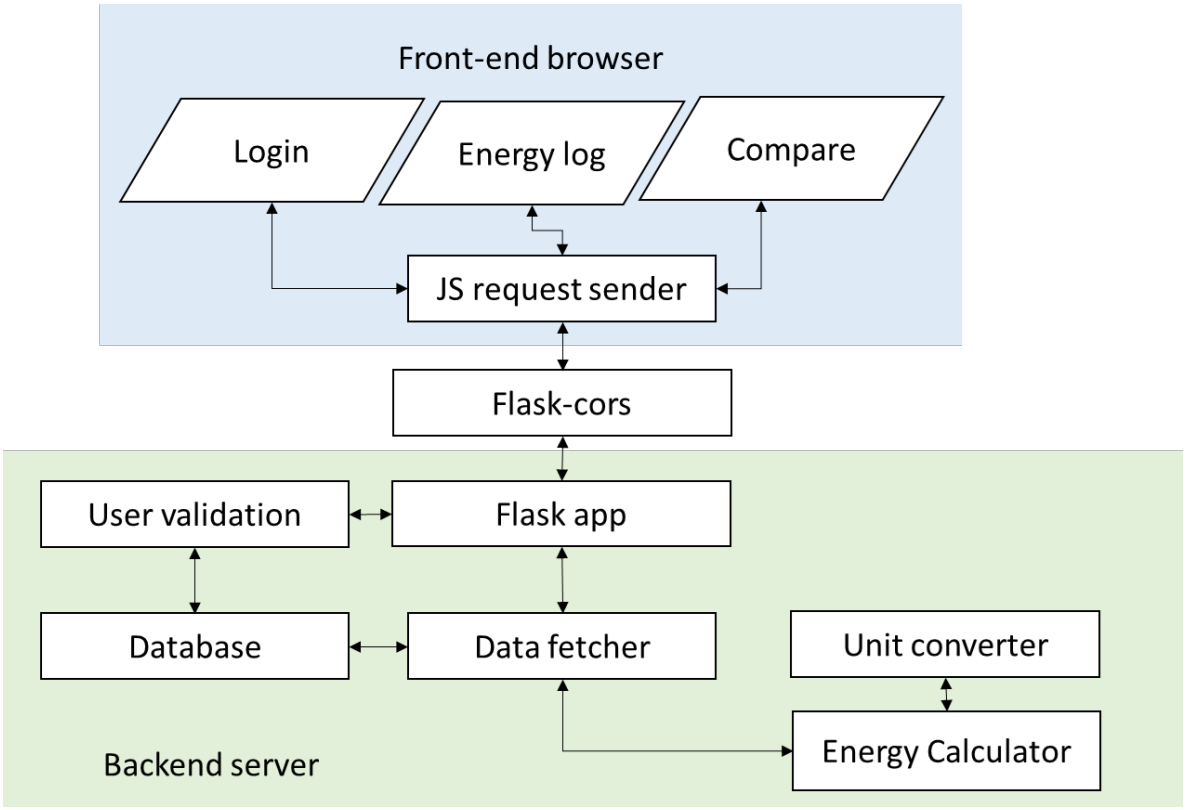

Figure 1: Diagram Showing an Overview

3.3 Unit Converter

The Unit Converter works hand-in-hand with the Energy Calculator system. This is what allows us to convert from Litres to kWh. This script is divided into three sections:

- **Litres to kWh**
- **Time Conversions**
- **SI Prefix Conversions**

The Litres to kWh section is currently being used in conjunction with the Energy Calculator script to produce the results we need in the correct format. The remaining two sections contain functions we may need in the future as the app is upgraded and expanded. They are not in use at this stage.

3.4 Front end

the Front end system is completely uncoupled from the rest of Power.Log to increase modality and security. The front end consists of three pages: index.html (which handles user login and sign up), log.html (which handles user log entries) and compare.html (which allows users to compare their energy consumption to others).
The front-end is built traditionally using HTML, JavaScript and CSS. This makes it easy to upscale and maintains because such front such end skill sets are common and the practices required are well documented.
The Chart.js library is used to aid in the drawing of graphs which help users visualize energy usage.

3.5 Server hosting and integration

The flask framework is used to host a python server via 'app.py'. While flask-cors facilitates communication between python and JavaScript via JSON objects. JSON is the favoured means of commu-

nication because it is an industry-standard, can be sent via the HTTP protocol, can be read by most programming languages and can be encrypted for imposed security.

3.6 Fuel Logging System

The Fuel Logging System consists of two scripts. The log.py script and the filePopulation.py script. These scripts work together to allow users to send and keep their fuel logs in the database.

3.6.1 log.py

This script is responsible for dealing with all the log data. It contains functions that can:

- Create a new log.
- Find and return the user's vehicle model.
- Return all the logs in a given file.
- Find a log by the date it was made.
- Find and return logs made over a certain time period.

This script essentially acts as a log management system. Inserting and retrieving any information that the user may need at any given moment/

3.6.2 filePopulation.py

This script is responsible for writing data into specified user files and reading data from files.

3.7 Data Fetcher

This system does what its name suggests and more. If any system in Power.Log requires any piece of data, the data fetcher will get it and send it where it is needed.

3.8 User Accounts & Encryption System

The userLogins.py script is a Python script that is responsible for handling information regarding users, specifically focusing on verification. It is currently implemented in the middle-end of the system. As such, the script is responsible for the following functionality:

- User Creation
- User Verification
- Password Encryption

The script is dependent on the 'filePopulation.py' module to read and write from the users' database. It also is directly connected to the front-end through JavaScript, which allows for the execution of middle-end processes as a result of changes from the front end.

The current encryption system is based on the idea of one-way encrypting. Instead of focusing on the decryption and encryption of passwords, our systems approach consists of taking the queried password, encoding it and comparing it to the existing, stored, encrypted password. This allows for safer encryption as passwords do not have to be decrypted, hence keeping it confidential.

3.9 Web Code

The Web Code is all the code used to create the front end of the website. This is all HTML, CSS, and JavaScript