Software Requirements Specification

for

Collect Forecast Visualize
(Central America)

Version 6.0

Prepared by Vitaliy Stepanov
Naoki Lucas
Andre Weertman
Max Ayala

WattTime

June 15, 2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Vitaliy Stepanov	11/11/20	New Use Case Diagram	2.1
Naoki	11/11/20	Added RTM	2.1
Vitaliy Stepanov	1/26/21	Removing satellite requirements	3.1
Max Ayala	1/26/21	Database structure finalized.	3.1
Max Ayala	2/9/21	Updated dependencies to reflect current forecast tools, added a reference to helpful material.	3.2
Andre Weertman	2/23/21	Updated 3.2.2 Visualization tools to match our transition from unity to react.	4.1
Vitaliy Stepanov	2/23/21	Updated Use Case Diagram and remove RTM.	4.2
Max Ayala	2/23/21	Updated 1.1 and 2.2, included high level diagram.	4.2
Max Ayala	3/16/21	Updated functional and non-functional requirements for forecast class.	5.0
Naoki Lucas	3/16/21	Updated functional and non-functional requirements for manager.	5.0
Vitaliy Stepanov	5/28/21	Updated all document to finalize requirements, added new UI images, removed raspberry-pi, and added constraint.	6.0

1. Introduction

1.1 Purpose

This project has two main goals. Firstly, this project will aid WattTime with their Climate Trace program. Climate Trace is an initiative to gather all the available greenhouse gas emissions and energy production data from across the globe into a single centralized source. This data will be analyzed to develop a real-time model that will help pinpoint major sources of air pollution from the electrical grid. This project will provide WattTime with four new data scrapers for countries in Central America, including Mexico, Costa Rica, Nicaragua, and El Salvador.

Secondly, the project will implement a separate application to visualize the data and forecast future values. The main purpose of this tool will be to gain further insight into the data and its potential implications in terms of pollution, as well as to learn new applicable skills in web development and time series forecasting.

1.2 Product Scope

A major problem we face today is the inability to verify the multitudes of self-reported data. Climate Trace will utilize new technologies to better track the sources of pollution. In return we empower investors to green. Furthermore, better data verification will mean accountability to industries that may not be up to code of the Paris Climate Agreements.

The overall goal of this project is to add functionality to Climate Trace. Additionally, these updates will directly improve the availability and capability of various eco-conscious technologies like WattTime's Automatic Emission Reduction (AER). In the long run, we endeavor to make Climate Trace a reliable authority on the most current emissions data. This tool will continue to be publicly open and free source to promote a greener future. We aim to inspire companies to develop energy efficient software, institutions to study the effects of our energy consumption, and even individuals to promote public awareness.

For a more detailed description you can check out this article co-authored by Al Gore, former vice president, and Gavin McCormick, founder and executive director of WattTime. If you have questions, please see the Press Release Frequently Asked Questions.

1.3 References

Gore, Al and Gavin McCormick. "We Can Solve the Climate Crisis by Tracing Pollution Back to it's Sources. A New Coalition Will Make it Possible." Medium, 15 July 2019. Online. <https://medium.com/@algore/we-can-solve-the-climate-crisis-by-tracing-pollution-back-to-its-sources-4f535f91a8dd>

Roberts, David. "We'll soon know the exact air pollution from every power plant in the world. That's huge." Vox, 27 May 2019. Online. <https://www.vox.com/energy-and-environment/2019/5/7/18530811/global-power-plants-real-time-pollution-data>

WattTime. WattTime, 2020. <http://watttime.org>

Pawar, Prathamesh. “Predicting Hourly Energy consumption of San Diego.” Medium: Towards Data Science, 7 June 2020. Online. < https://towardsdatascience.com/part-2-time-series-analysis-predicting-hourly-energy-consumption-of-san-diego-ii-f09665796c9>

2. Overall Description

2.1 Product Perspective

As an extension of an existing product, we are adding and utilizing scraper resources with WattTime. The application will make use of a database and create visual models for both real and forecasted data. This application, while in it of itself is new, does extends from the idea planted by WattTime. Our divergence from WattTime as our own product include features like being able to view our own Central American scrapped data further back in time and being provided a forecast for that data. With this comes our own database and tools that will be custom tailored to our pursuit, making this a self-contained product. The database will be made available to others with read-only access to specific points, and the application will have the ability to export data. This lets our users feel free to simply use or even develop off our platform.

2.2 Product Functions

Figure 2.2: *product backend*

Major functions of the product backend:

- Scrape websites for energy production data.
- Store website data in database
- Forecast energy production.
- Must be hosted in the cloud.

Major functions of product user side:

- GUI will display a heatmap that included the countries in our project and their percentage of renewable energy production.
- Export data from our database.
- Select the past and future data.
- Filter by pollution type, hour, and day.

2.3 User Classes and Characteristics

We have two predominant user classes. Primarily, our country specific scrapers will be used by WattTime affiliates as an expanded arm of their global reach for energy emission data collection. As a subclass to that group, AER compatible devices within the scope of our scrapers will implement our data into their reduction algorithms, lowering energy peak (dirty) times, therefore smart-device users within our country list by lineage will be users.

The second predominant class of our users will be the general public within the scope of our scrapers that have internet access. Even without AER compatible devices, anyone can reduce peak “dirty” energy hours by simple habit changes. Our API will give tools to the general public that allow them to see and forecast their own local energy power plants demand. This information can empower simple changes to personal energy consumption (such as starting dishwashers, and washing machines, or battery charging devices and the lowest energy hours). We assume this will predominantly be green energy enthusiasts, and those who care about climate change (which should be everyone by the way), but depending on local energy consumption charge rates, we may also offer money saving capabilities for those within power plants who charge differing rates across the cleanliness spectrum of their energy demand.

As a smaller class, users will be climate and energy researchers, those not specific to WattTime and its affiliates. Climate change is a world-wide effort, and our data addition to Climate trace may end up being a part of anyone’s effort to continue an energy map and forecast of the world. We make our data completely exportable in multiple formats, giving any research team free access to essential information in the continued fight against the most existential threat to us as we know it, climate change.

2.4 Design and Implementation Constraints

The main constraints for this project involve the difficulties in gathering available emissions data. Not every country has formal regulations for the reporting of this data, and as such many areas go unreported, or perhaps worse, report false data. Furthermore, if the data exists, it can come in a variety of languages and formats, such as spreadsheets, csv files, or just tables in a webpage. Some data will update in real-time (say maybe every 5 minutes), while other data will only update every hour or perhaps even every day.

This project is also non-profit and has realistically no budget. As such we will utilize free technologies, such as MongoDB. This will however constrain our space for storing historical data which will further limit the number of viable machine learning algorithms we can use. Also, this project is under the jurisdiction of WattTime and their policies. As such, we are constrained to developing in pep-8 format, and the output of the scrapers will have to be a list of dictionaries.

Lastly, the mongoDB limits us to 500 connections per 6 hours.

2.5 Assumptions and Dependencies

This project assumes that all data gathered from “reputable” sources is accurate. We know however, that this will not always be the case.

The data scrapers naturally assume that the format of the data will not change over time. A scraper will effectively be rendered obsolete whenever the data moves to a new website, comes in a new form, and is updated to include more information. Sometimes this will mean a minor fix and other times it will mean starting again from scratch.

Our application is heavily dependent on plotly library to visualize interactive data and a cloud hosting service.

Lastly this project will have various outside dependencies. Primarily these are the database used for storing historical data (MongoDB) and react for developing the visualization tool. Furthermore, there will be dependencies to each individual website from which data is being scraped.

Library dependencies include:

- plotly
- arrow
- bs4
- datetime
- requests
- selenium
- pandas
- pymongo
- pystan
- Fbprophet

3. Specific Requirements

3.1 User Interfaces

Our online API will be accessible globally to anyone with internet access. It will consist of an interactive page: map, data graphics, and data export. The home page will have an interactive globe that the user can spin and click on a country. In the sidebar on the left, historic and future graphs will display data based on the calendar of dates selected.

3.2 Functional Requirements

Functional requirements are provided in the following use cases, and graphically represented in our Use Case Diagram. Generally, our scrappers (that are also used by WattTime) continuously create raw data that fills our database.

3.2.1 Data Scrapers

Data Scrapers are specific to individual country API websites. They are meant to find hourly based energy usage. Some have entire day records provided and others we will run hourly timed scrapers across their data.

1. The data scraper must retrieve energy data from specific country websites.
2. The data scraper must return the data in dictionary format.

3.2.2 Forecast

1. The class must forecast future values of hourly generation for up to a week (168 total values per energy type per country)
2. Must be capable of cross validating data against the training set.
3. Will produce extra performance metrics, namely statistical properties, including r^2 and MSRE for verification.
4. Will provide capabilities for plotting forecasting data with the confidence intervals.

3.2.3 Manager

1. The database will automatically scrape data.
2. The database will check for missing entries.
3. The data can be re-requested to fill in missing entries.
4. Scrapers and Forecasters can run continuously.
5. Runs forecast's to generate new predictions monthly as data updates (except for Mexico).
6. Crashed scrapers/forecasts will be regenerated and re-attempted.

3.2.4 Visualization tool

This application will use the React library with JavaScript for data visualization.

Our user-friendly GUI will allow users to do the following:

1. Filter data by pollution type.
2. See past as early as 4 years back and forecasted data a week ahead.
3. Get specific country cleanliness meter, and hourly energy production by production type.
4. Access our map feature, and the data of the 4 countries we scraped by clicking their respective geographical locations.
5. Navigate between the different countries of our GUI fluently.
6. Export raw data.

3.3 Performance Requirements

3.3.1 Load

1. Up to 500 users in the past 6 hours may use the product, database, concurrently.

3.3.2 Storage

1. Database must store 4 years of historic data per country.
2. Database must store 1 week of forecasted data per country.
3. Robust. Back-up data.

3.3.3 Speed

1. Grab and format data from database for each country in 10 seconds
2. User actions will change GUI under 0.5 seconds.
3. Requests for data will be fetched and displayed within 2 seconds.
4. Export of currently displayed data should be in less than 5 seconds.
5. Load GUI in 5 seconds

3.3.4 Accuracy

1. Forecasting model should have r^2 value at or above 0.7

3.4 Logical Database Requirements

The database will be on MongoDB and on another machine the scrapers. Calling one a database server and another a scraper server, the scraper server will have scrapers constantly running on custom timers to collect data, at up to an hourly rate. Once this information is acquired, this scraper server then sends this data to our database. The scraper server will also do most of the maintenance for the database, such as removing old and unused data. The database will hold these data entries and should be consistent in both time and information when queried.

On the scraper server, since there will be a lot of idle time, the continuous machine learning algorithm will be computed on this machine as well. Once the algorithm is computed and the forecast is created, both will get sent to the database along with other useful data pertaining the training model. The next time a forecast/algorithm is computed, it will pull this data and any new datapoints from the scrapers for re-training. The server will also compare these forecasts to actual data as they eclipse and upload accuracy data per each power grid. The database will have portions of it made available with read-only access. The database is expected to run 24/7 and handle hundreds of queries a day at a minimum. As the project scales up, possibly a newer machine to handle thousands of queries an hour, however reliability amongst system components come first and this requirement is not severely bottlenecked by hardware.

3.5 Software Quality Attributes

Our system, with expansion in mind, is a flexible set of tools meant to apply to most/all countries. The database should readily accept data from any nation and the application should easily access this data. The only thing that might need extra work is how these objects are placed on a map. This means that as scrapers are built, the only concern should really be where on the map this data is presented. Scrapers will easily be interoperable and therefore maintainable. Anything presented to the user should be clear, consistent, and have useful links where applicable. Data should be presented to the user promptly when requested and the correctness and availability of data should be reliable.

The database will be on MongoDB since it will hold a crucial role for this project and should focus on being reliable and available 24/7. The scrapers and the GUI depend on this. The database should also hold any data that should serve useful/necessary by either the application or the scraper tools such as user export requests or an update to the iteration of forecasting models.