

Improving Accessibility to Solar Estimations for Non-Experts through a Flask-Based Online Platform

Christina ZORENBOEHMER¹

¹Z_GIS, University of Salzburg, Austria · christina.zorenboehmer@plus.ac.at

Abstract

With the rise of renewable energy sources, more and more homeowners are exploring the possibilities of installing their own solar panels. Those homeowners are consequently faced with the preliminary question “Would solar panels on my roof be worthwhile?” and historically, such an initial solar potential estimate has been marked by either technical or financial hurdles for non-experts. Many existing tools are integrated within commercial software packages or require extensive knowledge about the parameters and the execution of solar potential calculations. *Quick Solar Estimates* is a Flask-based web application that aims to make such calculations easily and intuitively accessible to any non-expert. It integrates a comprehensive python library dedicated to solar power calculations (pvlib), near-global meteorological data accessed (PVGIS), and an intuitive interface designed around an interactive map. The application additionally includes a SQLite3 database to optionally persist users’ solar estimations. In its current state, the lightweight application is deployed on a virtual machine on Google Cloud’s Compute Engine and offers an optimal setup for future expansion and scalability.

Keywords: Solar power, flask-based platform, web map, solar estimations

1. Introduction

1.1. Current Trends

Fuelled by prospects of climate change and the urgent need to reduce the consumption of fossil fuels there has been increasing political, economic, and social support and investment into renewable energy systems [1]. Of these, solar power stands out as the most accessible to individual homeowners who can mount photovoltaic (PV) installations on their own roofs or properties. Thanks to large-scale research funding and market demand for PV modules, their installations have become ever more feasible. This is especially true for residential PV systems as NREL [2] documents in Fig. 1.

PV installations are not only the most feasible residential source for renewable energy, but they are also highly effective. A 2013 study on residential solar panels in California found that 113,533 homes with PV modules avoided nearly 700,000 m³ of carbon dioxide [3]. The International Energy Agency (IEA) reports a steady increase in annual PV installations around the world [4] and in 2020, at least 139 gigawatts worth of PV modules were newly installed worldwide (see Fig. 2). For reference, the cumulative wind power generated in all of Europe in 2014 summed to 133 gigawatts. The total solar power harnessed from PV installations thereby hit approximately 760 gigawatts in 2020 (Fig. 2). Data from 2021 also shows that the rate of annual rooftop PV installations is increasing. With a total of 173,000 terawatts of solar irradiance reaching the Earth’s surface at any given time, which

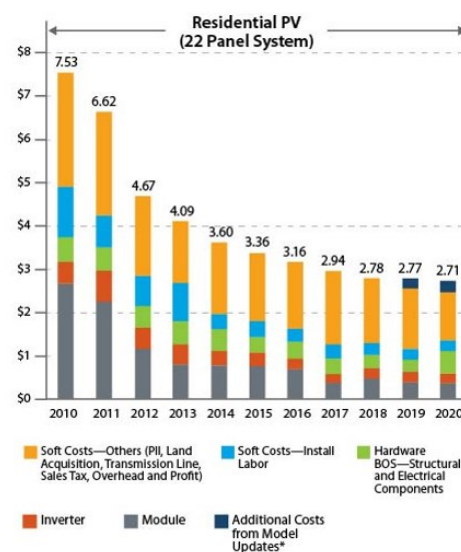


Figure 1: Costs of PV systems 2010 - 2020. Source: [2].

approximates to more than 10,000 times the energy usage of the world, the remaining potential for solar energy harvesting is immense [5].

1.2. A Need for a Non-Expert Tool

With regards to these trends, it becomes clear that there is an increasing

interest in the acquisition of PV modules. As such, homeowners are consequently faced with the preliminary question “Would solar panels on my roof be worthwhile?” and historically, such an initial solar potential estimate has been either technically or financially difficult to access by non-experts. A variety of existing tools [6, 7, 8] are structured sub-optimally for a simple query that may answer the above stated question. For example, solar power estimation tools are often integrated within software packages that users must first download, install, and potentially pay to use. A desktop software also encompasses a much larger and more detailed scope of analysis than would be required for a first solar potential estimation. Likewise, many online tools are far more extensive, and thereby potentially overwhelming, for newcomer PV customers. This goes hand in hand with the requirement of a baseline knowledge about the input parameters such as the PV azimuth, ground tilt, or even power conversion mechanisms. The output format of the analysis results is most commonly in the form of written text or tables, leaving the interpretation up to user. Another alternative is a paid consultation with expert companies.

These solar estimation methods shine light on an existing gap in the available tools and a need for a free, simplified, and visually intuitive tool, where any non-expert is able to configure and analyse their own solar estimation. This is the central aim and motivation behind the creation of an online *Quick Solar Estimates* application.

2. Methods

2.1. Overview

Quick Solar Estimates is a Flask-based web application designed for desktop usage where users may intuitively calculate a simple solar estimation in the form of irradiance (Watts per m²). The application is designed to help fill the aforementioned gap among available tools for non-experts. The application broadly consists of two broad structural pillars:

- 1) the application’s infrastructure as the structural framework, and
- 2) the solar estimation calculations as the thematic content of the application.

As a whole, the web application integrates an interactive, map-based user interface with modules by the open-source python package pvlb [8], which are designed for comprehensive solar power modelling, and near-global meteorological data from PVGIS [9]. In the following sub-sections, the design and functionalities of *Quick Solar Estimates* are described and discussed in detail.

2.2. Infrastructure

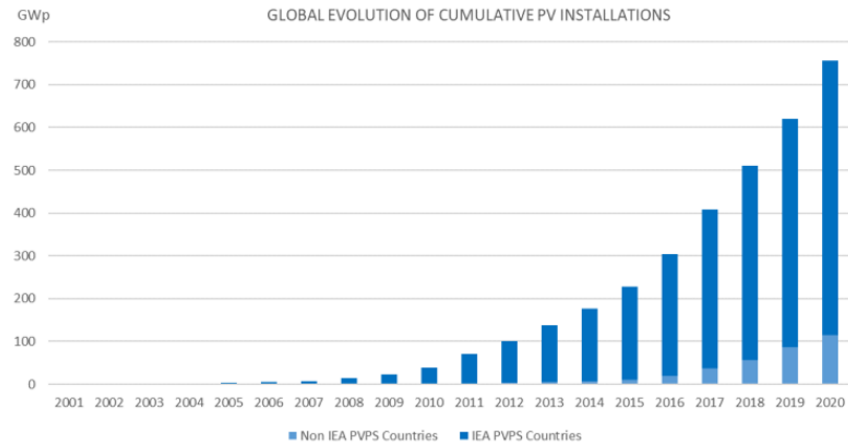


Figure 2: Global Evolution of Cumulative PV Installations. Source: IEA PVPS [4].

Quick Solar Estimates was developed in the PyCharm Professional (2021.3.2) Integrated Development Environment (IDE) in the form of a Flask project. The application was tested and debugged on a localhost deployment, before it was containerised as a Docker image for online deployment on Google Cloud. The complete source code for the application as well as a video demonstration can be found on GitHub: <https://github.com/Christina1281995/qse>.

2.2.1 Infrastructure Component: HTML, JS, CSS Frontend

The frontend of the platform consists of HTML5, JavaScript, and CSS files. The user interface is designed around the main, landing page within which an interactive OpenLayers [12] web map is the central element (shown in Fig. 3). The map, programmed in JavaScript, is designed to help users enter location and rotation parameters of a hypothetical PV panel in a visual and intuitive manner, as opposed to having to enter numeric values for the coordinates and panel rotation. The initial status of the map features a worldwide OpenStreetMap base map, though three further base maps are optionally available via a layer switcher. Users can navigate the map either by using their mouse or through a search box, which implements a MapQuest geocoder [10]. Once the user has found their home or any alternative point of interest, they simply click on the map to drop a red, rectangular geometry representing the solar panel. Simultaneously the base map changes to a Google Satellite base map and zooms to a house-level scale. These functions are based on the assumption that users will visually recognise their own roof in a zoomed-in satellite image. The satellite image will then provide a basis with which the users can rotate the red, hypothetical PV panel according to the now visible shapes of the roof.

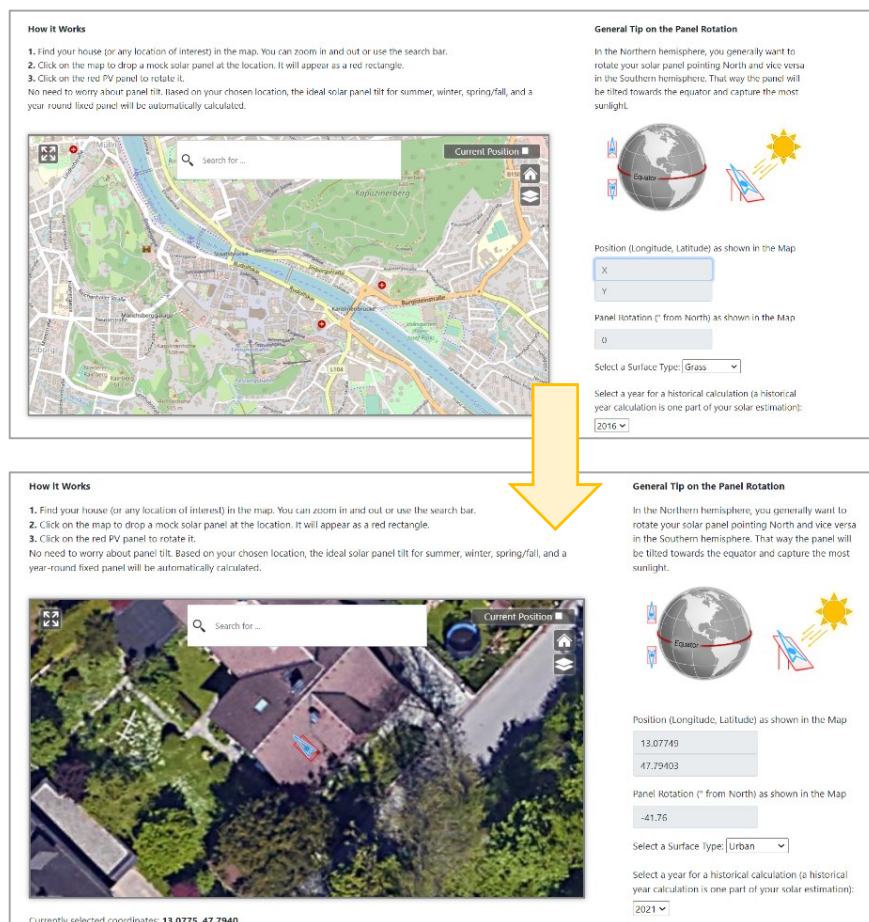


Figure 3: The main landing page of *Quick Solar Estimates*. Top: the initial map view. Bottom: the map view after the user clicks on their home and rotates the red geometry (representing a solar panel).

As the user interacts with the map, a bootstrap form adjacent to the map is automatically filled with the corresponding numerical coordinate and rotation values. These form elements are set to "read only", meaning the user can view the numeric values but not manipulate them in the form (see Fig. 3). A simple set of instructions and hints are also provided on this main page to offer an appropriate level of support. In particular, these hints inform the user how the panel rotation affects the solar irradiance through a simple graphic, which shows which end of the rotation arrow is tilted upwards.

Further input parameters are selected from predefined drop down boxes. Since all input types are either map-based or drop-down boxes, they ensure that the input is correctly formatted and avoid the risk of non-conform text-based input, which may lead to issues with data formatting. It also helps avoid any database security risks such as injection attacks or insertion errors.

Other web pages include a login and signup page, a results page where outputs are displayed in a graphical manner to help the user visually interpret the results, and a feedback page consisting of ranking and text based questions, and a page for loading saved searches in a table format. Aside from the main page, the results page contains a more elaborate design. After the user clicks “calculate” on the main page, the results page is rendered with the immediately, graphically visualised calculation results. In the current application deployment the results page is loaded synchronous with the created graphs, which means the page loads for a few seconds before appearing with all contents rendered. Explanatory texts accompany the graphs with dynamically inserted values such as the year of the data or certain irradiance values. These texts serve as an aide to help the user review the used parameters and the outputs. The results page additionally allows the user to save their search as a user account and to download the results as a pre-formatted PDF. For this, the JavaScript library “jsPDF” [13] was used.

Quick Solar Estimates

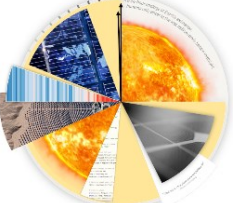
About QSE

User Feedback

Sign Up

Login

Create an Account to Save Any of Your Searches



Quick Solar Estimates

Saved Searches

About QSE

User Feedback

Log Out

Feedback

By filling in this form you will assist the improvement of this platform. Any constructive feedback is highly appreciated.

1. How would you rate your overall satisfaction with this platform?

★★★★★

2. How would you rate your interaction with the map?

★★★★★

3. Were you able to understand how to use the platform intuitively?

★★★★★

4. Were the results clear?

★★★★★

5. Would you recommend this platform to others?

★★★★★

Solar estimations by [Potsdam](#) and [Globe](#)

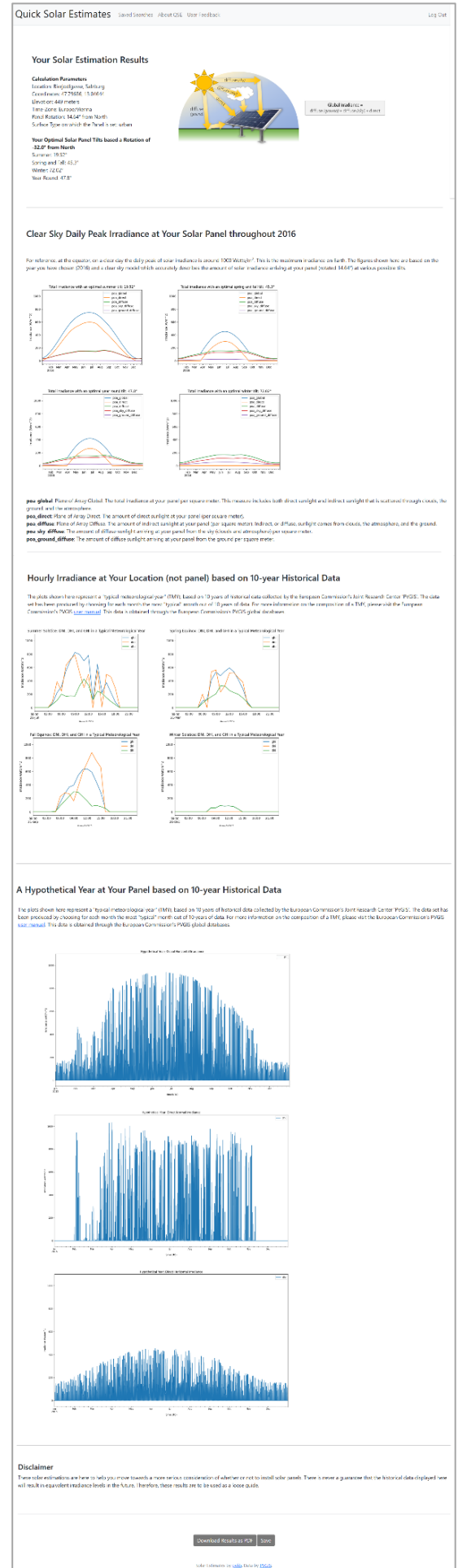


Figure 4: Right: The results page, rendered with an example result set. Left-top: The sign up page. Left-bottom: The user feedback page.

2.2.2. Infrastructure Component: Python Backend

The backend and main logic underlying the platform is designed according to a Flask framework [14] wherein the routing and template rendering is performed via customised python functions, collectively stored in a “main.py” file. These functions are called when a new web route (e.g. “/login”) is requested via http GET or POST methods. Aside from the main.py file, containing the logic for routing and database connections, two further python files contain the functions used to operate the platform and calculate solar estimate results.

Flask is a micro web framework, which offers a convenient layout and format for application designing. Since the *Quick Solar Estimates* application does not contain any extensive connections, libraries or tools it was deemed an appropriate programming approach. With Flask it is possible to retrieve and send data to and from various HTML elements in the frontend, which conveniently allows for python-based interaction with the input data, a calculation of solar potential via sophisticated python packages and a graphical result generation. Those generated results are then passed back to the frontend in a pre-formatted template.

2.2.3. Infrastructure Component: SQLite Database

As a third infrastructural element, a lightweight SQLite3 database is integrated to keep track of user, feedback, and saved search data. Since the currently deployed version of *Quick Solar Estimates* does not require the storage of spatial data types and only three database tables are used, a small-scale relational

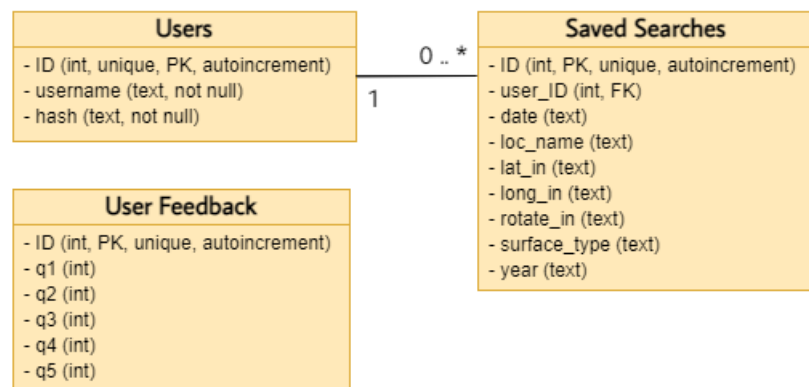


Figure 5: Conceptual Schema for the SQLite3 database.

database system was chosen. A conceptual database model is displayed in Fig. 5. To ensure password protection, a python package “werkzeug” is utilised to hash users’ passwords. Additionally, injection attacks are prevented by limiting the options for text-based input and by using a parameterised query. Data is queried and updated using custom functions in Standard Query Language (SQL) as in the following example:

```
# Function for inserting data into db
def insert_db(query, args=()):
    con = sqlite3.connect('database.db')
    cur = con.cursor()
    cur.execute(query, args)
    con.commit()
    con.close()
    return
```

```
# Add feedback to database
insert_db("INSERT INTO feedback (q1, q2, q3, q4, q5, comment) VALUES (?, ?, ?, ?, ?, ?);",
        (fb[0], fb[1], fb[2], fb[3], fb[4], comment))
```


2.3. Docker and Google Cloud Deployment

The finalised application was converted into a Docker container image and uploaded to a public repository on DockerHub (<https://hub.docker.com/repository/docker/christina1281/qse>). With a paid account on Google's Cloud platform, a container-optimised virtual machine (VM) was instantiated on Compute Engine. Compute Engine is a form of Infrastructure as a Service, with which a complete operating system is instantiated and can be used to virtually run the application online. Once the VM was created, the public Docker image from DockerHub was pulled into the VM via Secure Shell Protocol (SSH) and run. This process results in the live deployment of the application, accessible via an IP address. As a last deployment step, this IP address was mapped to a previously owned domain "zorenboehmer.com" to give the final result of a live application on the web address <http://qse.zorenboehmer.com>.

2.4 Solar Potential Estimations

The second structural pillar, next to the infrastructure of the application, is the content-related solar potential calculation. This pillar represents both the logic and purpose behind the application. A combination of solar modelling functionalities by pvlib, the user inputs, and various meteorological data by PVGIS is used to generate three sets of results:

- 1) clearsky daily peak irradiance (Watts/m²) at the hypothetical PV panel throughout a user-selected historical year,
- 2) hourly irradiance (Watts/m²) at the user-selected location based on 10-year historical data,
- 3) a hypothetical year at the user-selected hypothetical PV panel based on 10-year historical data.

As mentioned previously, the results for these three categories are returned to the user in a graphical format, accompanied by explanatory text. In the following subsections more detail is provided about the utilised python packages, data sources, and their functions.

2.4.1. Pvlib

To perform a basic set of solar estimations, *Quick Solar Estimates* uses an open source, state-of-the-art python package pvlib. Pvlib is a community-supported tool, first published in 2015, that provides a set of functions and classes for simulating the performance of PV energy systems. In this project, the pvlib functionalities were first used to create a location object containing coordinates, a time zone reference, a location name, and an altitude (meters above sea level). Using this location object, and pvlib-internal solar position data, a whole year of clearsky irradiance could be calculated based on the user's chosen year from a drop-down box (available years are 2016 – 2021). The clearsky model is based on the 'Ineichen and Perez' clear sky model [15], which is based on the atmosphere's turbidity, i.e. the optical thickness of the atmosphere due to water vapour and aerosol particles. The required turbidity data is provided within the pvlib package. Since this calculation generates clear sky values, the resulting graphs appear smoothed and without the irregularities and outliers typical of weather-related solar irradiance. These results offer a good first estimation on the hypothetical solar potential at a PV panel at the user's chosen location.

The following two result categories are not generated with pvlib internal datasets but with decades' worth of historical meteorological data collected and provided by the European Commission through their photovoltaic GIS 'PVGIS'.

2.4.2. PVGIS

To calculate hourly and yearly irradiation values for the user's selected location, a typical meteorological year (TMY) is required. A TMY is a standardised (ISO 15927-4), hypothetical year, which averages 10 actual, historical years. It consists of hourly data for solar radiation, temperature, air mass, wind speed

and other meteorological data, obtained from satellite imagery and ground measurement stations, which is coerced into a single “typical” year. This coercion process is done by comparing all of the same months (e.g. all Januaries) throughout the 10 years, and selecting the most representative one, whereby ‘representative’ refers to the most average temperature and air mass

values. Alongside the high temporal-resolution of the data, PVGIS operates on four databases that together offer a near-global coverage (see Fig. 6). Depending on the provided coordinates, the most appropriate database is automatically selected to query and return TMY data. This data can be programmatically accessed directly through pvlib using a HTTP GET method wherein location and timespan parameters are dynamically inserted into the request.

Once this data is obtained, the next two result categories can be calculated and visualised. Based on the TMY data, a pvlib function for the total irradiance, I_{tot} , at the hypothetical PV panel is called. The function combines I_{beam} , which is direct solar irradiance arriving in a perpendicular beam to the PV panel, $I_{skydiffuse}$, which is atmospherically scattered irradiance arriving at the PV panel, and I_{ground} , which is irradiance scattered from the ground surface:

$$I_{tot} = I_{beam} + I_{skydiffuse} + I_{ground}$$

The individual components of this function are each separately calculated by pvlib based on the location object, TMY data, elevation measures and the surface type. The surface type is a further user input, which is selected from a drop-down box on the main page.

2.4.3. Further Functionalities and Parameters

Since accurate solar calculations often require a myriad of input parameters, even beyond those already discussed and obtained through pvlib, PVGIS, and the user input, a few additional libraries and API access points are necessarily integrated into *Quick Solar Estimates* as further data sources. The first of those is Google’s paid Elevation API, through which worldwide elevation data can be obtained at any given coordinate point. The altitude in meters above sea level is a mandatory input parameter to all solar calculations. The implications and considerations of using the ground level altitude are later discussed in section ‘4. Discussion’. A second additional function is a custom-built PV panel tilt generation method. The panel tilt, in degrees relative to horizontal, is another key parameter that is mandatory for many of the pvlib calculations. Rather than using merely one tilt parameter, a method by SinoVoltaics [11] is used for deriving optimal panel tilts for each season and around the year based on the user’s chosen latitude. Lastly, a MapQuest reverse geocoder is implemented to provide a location name for the results and for the pvlib location object. This parameter is more for aesthetics than for functionality since it is not required as a mandatory input for the result generation. It is, however, used to give a title to the PDF output. With all infrastructure and methodological elements in place, *Quick Solar Estimates* can take any users’ inputs and generate a structured set of results.

Coverage of Solar Radiation Databases

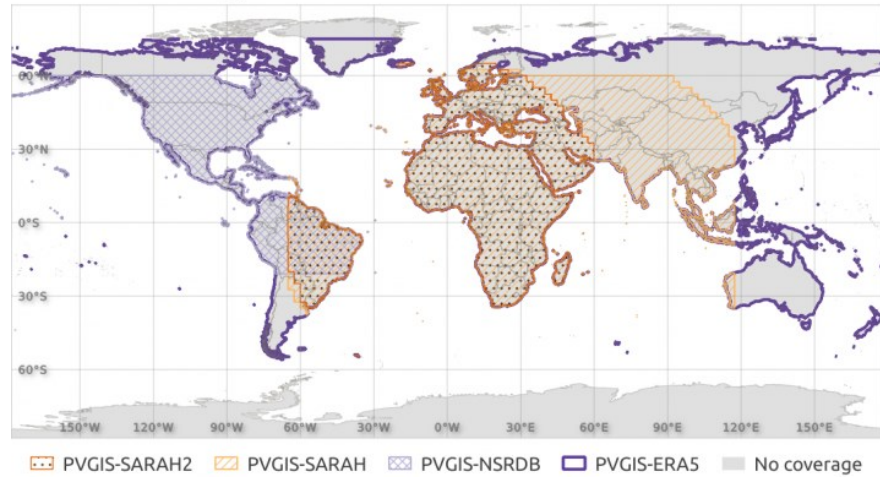


Figure 6: Coverage of Solar Radiation Databases. Source: European Commission [9].

Required Parameters	Method for Obtaining Parameter
Latitude and Longitude	User input via the interactive Map
Rotation (° from North)	User input via the interactive Map
Surface Type	User input via pre-defined drop-down box
Historical Year to use for Clearsky Calculation	User input via pre-defined drop-down box
Panel Tilts (° from horizontal) for Summer, Winter, Spring/Fall, Year Round	Calculated using Latitude
Altitude (meters above sea level)	Google Elevation API using Latitude and Longitude
Location Name	MapQuest Reverse Geocoding using Latitude and longitude
Pvlib.location class instance	Created with pvlib using latitude, longitude, rotation, altitude
Air Mass	Calculated with pvlib using the pvlib.location object
Solar Positions (Zenith, Azimuth)	Calculated with pvlib using the pvlib.location object
TMY (typical meteorological year)	Accessed through PVGIS using latitude and longitude

Table 1: Parameters required for the Quick Solar Estimates application and the corresponding methods for obtaining them.

3. Results

With the aforementioned accumulation of input parameters (see Table 1) and methods for obtaining a solar estimation, *Quick Solar Estimates* generates the three result categories listed in '2.4. Solar Potential Estimations'. The clearsky results amount to four individual graphs, whereby each corresponds one of the four previously calculated panel tilts. Since the tilts are optimised for summer, winter, spring/fall and year round, the clearsky results show how the irradiance at the panel would differ based on which panel tilt is set (see summer tilt example in Fig. 7). The hourly result set also amounts to four individual graphs, whereby these graphs represent the summer and winter solstices, and the spring and fall equinoxes and show hourly irradiation (Watts/m²) based on the TMY. For an example, see Fig. 8. Lastly, a whole hypothetical year of irradiance is generated based on the TMY data and returned to the user in three graphs showing the direct, diffuse and global irradiance respectively (see Fig. 9 for an example). In addition to viewing these results directly in the browser, they can be downloaded as a PDF or added to the users' account to access the parameters again at a later time for re-calculation.

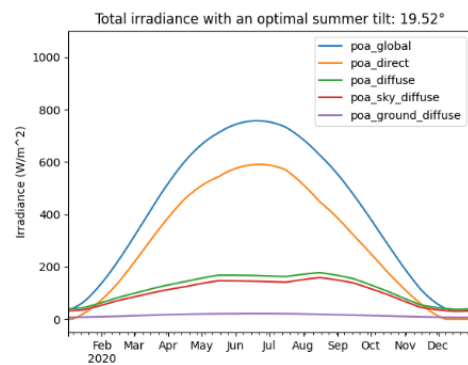


Figure 7: Summer Clearsky Irradiance example.

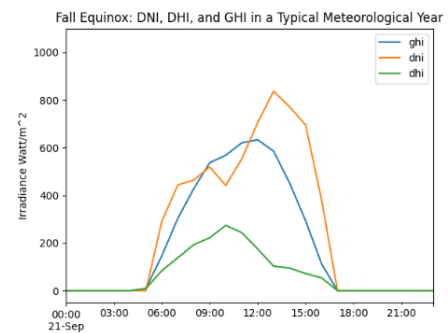


Figure 8: Hourly Fall Equinox Irradiance example.

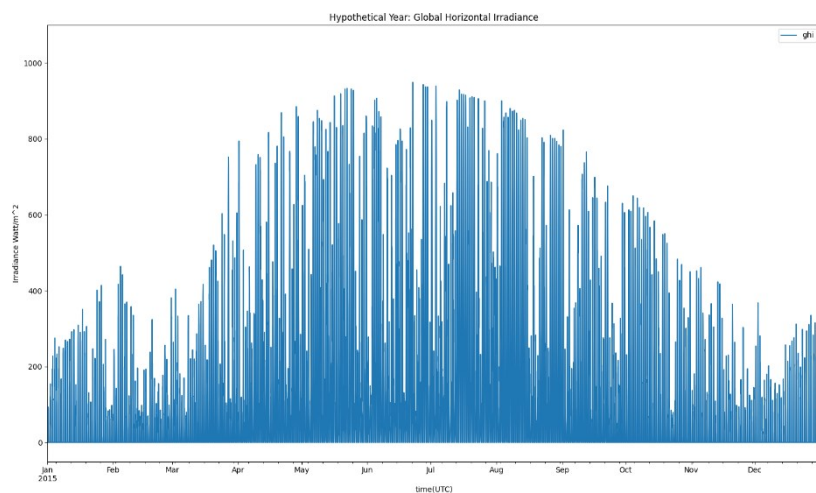


Figure 9: Hypothetical year global irradiance example.

4. Discussion

Firstly and foremostly, the overall utility and usefulness of the results must be discussed. Since this first version of *Quick Solar Estimates* is a minimalistic application that merely generates irradiance values, it cannot offer the user indications on the amount of power that might be generated. For this, various additional parameters would be required, such as the size and number of PV panels, the specific type of panel and its power conversion rates. This shortcoming may be overcome in the future, by implementing an expanded version of *Quick Solar Estimates*.

Regarding the calculation of optimal panel tilts around the year, there are large numbers of online sources proposing different methods. These range from 'rules of thumb' to complex algorithms requiring various solar parameters. For the purposes of the Quick Solar Estimates platform, a comparatively straightforward method by SinoVoltaics based on latitude was chosen but not yet validated as either accurate or inaccurate. Further investigation and validation activities should be conducted in the future.

Lastly, there are obvious shortcomings related to the altitude data obtained from Google Elevation API, which only returns the elevation on ground level, which may lead slightly inaccurate pvlib estimations on temperature, wind speed, or air pressure. This data could possibly have been replaced in some states by high-resolution digital surface models (DSM). DSMs however, are very large in terms of their required storage space and not freely or consistently available to the same extent as the PVGIS databases. To work with selected DSM data would therefore have improved the results' accuracies but it would, at the same time, have drastically limited the geographical areas where Quick Solar Estimates could be applied. It would therefore have represented a considerable bottleneck to the application and was therefore, for the time being, not opted for.

5. Outlook

In its current state, the *Quick Solar Estimates* platform represents a first step towards a comprehensive non-expert solar estimation tool. Though its current functionalities are kept lightweight in terms of the extent and complexity of the solar calculations, it is equipped with a powerful setup consistent of the extensive pvlib python package, a connection to the near-global PVGIS meteorological data, and a database that can persist users' data. This combination represents an optimal setup for future expansions to the application since all three components offer a wide range of yet unused resources for more extensive solar estimations.

Looking ahead, concrete examples of scaling up the application's functionalities include direct comparisons between two or more points of interest (i.e. hypothetical solar panels), the calculation of power output from a specified area of PV panels, more sophisticated panel tilt recommendations, and further processing of the current Google Elevation API results which return the ground elevation above sea level. In terms of aesthetics and interface design, future expansions could include the integration of map screenshots in the results, which show the user's hypothetical solar panel on a satellite imagery base map. Such screenshots may be useful as a visual reminder of the exact location and rotation of the modelled results, and have previously been attempted to generate using the JavaScript library 'html2canvas', however the use of Google's base maps prevents the download, storage, or further work with the generated screenshots. Another interface improvement may be the switch to an interactive result presentation, rather than the current static display of graphs and values. To achieve this the use of JavaScript library 'mpld3', which is an extension to matplotlib. Any future work on the *Quick Solar*

Estimates application may include any of the above mentioned, but is by no means limited to only those examples.

Conclusion

All in all, Quick Solar Estimates offers non-expert users an easy access to a first solar potential estimation by generating three sets of results on historical and hypothetical irradiance at a selected location. The platform's main aim is to offer a simple user interface where the input mechanisms are intuitive. The platform successfully combines various infrastructural and methodological components to achieve a near-global coverage for its analyses. Looking towards the future, there is great potential for upgrading and expanding the platform to allow for more inputs and more sophisticated outputs.

References

- [1] Kabir, E., Kumar, P., Kumar, S., Adelodun, A.A. and Kim, K.H., 2018. *Solar energy: Potential and future prospects*. Renewable and Sustainable Energy Reviews, 82, pp.894-900.
- [2] National Renewable Energy Laboratory (NREL), 2021. *Documenting a Decade of Cost Declines for PV Systems: NREL Marks Ongoing Cost Reductions for Installed Photovoltaic Systems, While Also Establishing Benchmark of PV-Plus-Storage Systems*. [Online]. Available at: <https://www.nrel.gov/news/program/2021/documenting-a-decade-of-cost-declines-for-pv-systems.html>. (Accessed: 28.03.2022).
- [3] Arif, M.S., 2013. *Residential solar panels and their impact on the reduction of carbon emissions*. University of California, Berkeley. Retrieved from https://nature.berkeley.edu/classes/es196/projects/2013final/ArifM_2013.pdf.
- [4] International Energy Agency Photovoltaic Power Systems Programme (IEA PVPS), 2021. *Snapshot of Global PV Markets 2021*. Report IEA-PVPS T1-39: 2021, Technology Collaboration Programme.
- [5] Chandler, D.L., 2011. *Shining Brightly: Vast amounts of solar energy radiate to the Earth constantly, but tapping that energy cost-effectively remains a challenge*. [Online]. Available at: <https://news.mit.edu/2011/energy-scale-part3-1026>. (Accessed: 28.03.2022).
- [6] Schmidt, P., 2022. *Test Your Home's Solar Power Potential*. [Online]. Available at: <https://www.storey.com/article/test-home-solar-power-potential/>. (Accessed: 28.03.2022).
- [7] EasySolar, 2022. *EasySolar solar software: the easiest way to design photovoltaic systems*. [Online]. Available at: <https://photovoltaic-software.com/easysolar-solar-software-the-easiest-way-to-design-photovoltaic-systems>. (Accessed: 28.03.2022).
- [8] Holmgren, W.F., Hansen, C.W. and Mikofski, M.A., 2018. Mikofski. *pvlb python: a python package for modeling solar energy systems*. Journal of Open Source Software, 3(29), 884, <https://doi.org/10.21105/joss.00884>
- [9] European Commission, 2022. *PVGIS Photovoltaic Geographical Information System*. [Online]. Available at: https://joint-research-centre.ec.europa.eu/pvgis-photovoltaic-geographical-information-system_en. (Accessed: 28.03.2022).
- [10] MapQuest Developer, 2022. Geocoding API. Available at: <https://developer.mapquest.com/documentation/geocoding-api/> (Accessed: 03.06.2022).
- [11] Rooij, D.D., 2015. *Solar Panel Angle: how to calculate solar panel tilt angle?* Available at: <https://sinovoltaics.com/learning-center/system-design/solar-panel-angle-tilt-calculation/> (Accessed: 15.06.2022).
- [12] OpenLayers, 2022. *OpenLayers*. Available at: <https://openlayers.org/> (Accessed: 30.06.2022).
- [13] Hall, J., 2021. jsPDF. Available at: <https://www.npmjs.com/package/jspdf> (Accessed: 12.06.2022).
- [14] Grinberg, M., 2018. *Flask web development: developing web applications with python*, O'Reilly Media, Inc.
- [15] Ineichen, p., and Perez, R., 2002. *A new airmass independent formulation for the Linke turbidity coefficient*. Solar Energy, 73, pp. 151-157.