**Satellite Shade Analysis**

**Final Document**

 **Prepared for: Seth Holland**

**Prepared by: JBOP Solutions**

Taylor Born,

Ben Johnson,

Steffen Olson,

Rachel Pang

Version 1.00

Last Updated: April 30th, 2014

Table of Contents:

Introduction 1-3

Process Overview 3-4

Requirements 4-8

Architecture 8-10

Testing 10-12

Future Work 12

Conclusion and Glossary 13

# Introduction:

Electrical power has become an ever more important resource in our modern, information-based economy, with average household consumption rising steadily to power an ever-increasing array of high tech devices. Up until recently, people have relied solely on power plants powered by fossil or nuclear fuels to provide power to their homes, which contribute to pollution, waste and ecological impacts. Solar power has been a growing trend in providing clean, reliable energy for homes and businesses, and since its introduction in the 1970’s, it’s now more affordable and more accessible for everyday people to invest in. Because of this, interest in solar installation has skyrocketed.

Rooftop Solar is one of these solar paneling companies that is located in Flagstaff, Arizona that deals with the sales, marketing and installation of solar panels on buildings. On average, they receive around 30-40 clients a month that may end up being potential adopters of solar energy, and give them the information and estimations needed to make a decision on whether it is right for them. They are just one of hundreds of the companies throughout the United States involved with the effort to make energy production greener.

## The Current Process of Shade Estimation

There’s a lot of information to take into account when estimating the solar energy one would receive from paneling. There’s basic knowledge required such as the angle of the roof, as well as the amount of solar paneling desired. However, the hardest part of the estimate relies on obstructions around the building, such as trees and other buildings. 

To estimate this shade, a device called a SunEye (see fig. 1) is used to take an upwards fish-eye lens photo of where the panel plans to be installed, and is done where every panel would be installed to capture any surrounding obstructions. This data is then ran through an algorithm that uses sun positioning throughout the course of a year to estimate how much sunlight hits those panels (in a percentage) over the entire year. The amount of shade that hits the panels can make or break an investment of solar energy, with a bad estimate possibly costing an adopter more money than they would recoup from the solar paneling.

## Our Alternative to Performing Shade Estimations

Our selected capstone project deals specifically with the inefficiencies of the current methods solar companies around the world use in order to get an initial shade analysis estimate of a client’s roof. The current process involves sending employees on the roof of clients’ buildings to do the analysis with the SunEye. This is not only a huge use of time, as it requires employees to go on site to every potential contract, but a safety hazard for employees, as they need to climb up on every roof and perform the solar eye analysis.

Our project allows the company to perform these analyses through a web application. This application will take in an image from Google Maps, and allow the user to mark where obstructions are seen in the image, along with where the solar panels plan to be placed. The application will then use this information, with their relative heights and the image scale, and calculate how much percentage of shade will cover the panels over the course of a year (see fig. 2 for an screenshot of our front-end). Right now, our solution provides results very close to the SunEye and can serve as a very viable alternative to performing initial estimates.

Besides the calculation application, our project also encompasses a variety of extra features which includes a secure user system, which allows for logging in with a Google Account, and a project creation system that includes information about the locations (client name, address, extra information) and the locations of each of the user marked obstructions and solar panels in the images.

To keep track of all this information accurately, we are using databases implemented through Google App Engine; this will store information related to the current user logged in, as well as the information stored on each of the projects created. This project is contained on a Google appspot website that we will create and will eventually be directed to on Rooftop Solar’s website for them to license out to other companies later on.

## Our Created Alternative's Goal

This application will give an accurate estimate on how much shade will hit the panels throughout a year, solar companies will be able to give this information to potential clients/buyers to make an educated decision on if solar panel installation would be appropriate for their home. This would lead to a contract signing for the purchase and installation of solar panels, or save time in not having to send an employee on site to perform a shade estimate.

With future plans from Rooftop Solar to monetize and license out the application to other companies in the future, this application has the potential to save the time of solar companies around the globe in performing an accurate shade estimate, while also saving money and preventing accidental employee injury from climbing up on more rooftops.

# Process Overview

The overall process of creating our project was a combination of a waterfall and iterative method. In the first semester, we created requirements documentation and we planned our system by explaining it through presentations we gave to the class in our first semester of the project. In the second semester, we adopted a more iterative method during the implementation stage. We used Bitbucket and Sourcetree for our project to save and edit our code. This was the source of and the tools for our iterative process as we could make changes to our project by adding or reverting back to previously saved commitments. This kept our project flexible which was important since this project has only been done by one other company-- and none of their information is available outside of the company.

Ben was our team leader, document manager, and mentor contact. He coordinated the tasks we were to complete for the week, as well as drafting up documentation to present to Dr. Doerry every week that outlined what we were planning to do for that week, what was completed the previous week, and what still needed to be done. Ben was also in charge of pushing changes to the public website. He did work where needed, such as various tasks on both the front and back end of the website, as well as completing and formatting major documents, such as the requirements and design document before delivering them to Dr. Doerry and Rooftop Solar.

Rachel was in charge of being the sponsor contact. She helped with some of the front end work, however most of her work lied in documentation and presentation creation. Rachel completed most of the work on all of the Powerpoints for the presentations, as well as the final poster for the Undergraduate Research and Design Symposium. She also helped Ben in completing major documentation that went to our sponsor and mentor.

Taylor was in charge of the implementation of the algorithm and working with the python files to make them work. He met with our sponsor along with Steffen to make sure that the algorithm of how the shade percentage is calculated along with the sun’s position in various places on the equator. He also did some work on the database in relation to projects and users to make sure that the basic functionality of creating and accessing projects and user accounts worked properly.

As previously mentioned, Steffen worked with Taylor on a lot of the backend of the project. Steffen worked with Taylor on some of the algorithm implementation as well as helping with the creation and access of projects and accounts. Steffen’s main work was in integrating the website with Google application such as maps and accounts. He worked on the Google maps interface as well as the Edit Projects page to make sure that the fields in the Edit Projects page worked seamlessly with Google maps to find the location.

# Requirements

For our requirements, we started by speaking with our sponsor, Seth Holland, co-founder of Rooftop Solar on what he expected from, and wanted to see in our application design. After a few meetings, we begun drafting the requirements according to his specification. There were three main sets of requirements considered, including functional, environmental and nonfunctional. This section will contain a summary of each of these as well as details relevant to each section.

## Functional Requirements

Functional requirements for this project focused primarily on all the major features of our application. The main section involved user functionality, including integration with Google Accounts, the creation of a project system, as well as the ability to mark-up a Google Maps image and calculate the sun access. In addition, this also includes the tools that we plan to use to create this functionality, such as the Google App Engine, Google Maps and Google Draw APIs.

1. User Functionality
	1. User Accounts
		1. Log in/out system - Users can log into or out of their Google Account using username and password
	2. Project Creation
		1. Name of project - encompasses a user input for reference in project list at a future time.
		2. Client name - User can input a name for the client to keep reference on who the specific project is for
		3. Location input - inputs address in that specific region to import from Google Maps for use
	3. Project Loading/Saving
		1. Project Database loaded that encompasses that user’s current projects and the ability to select one of them, or creation of a new project
		2. Projects are saved upon creation, as well as whenever the user desires to save the project, where it will also contain the coordinates of every obstruction and module, along with their heights in the database.
		3. Loading will allow for any saved obstructions or modules to repopulate a Google Maps image with coordinates of each of them, along with their heights
	4. Google Maps Image Modification
		1. Image is imported and initially marked with current obstructions seen in the imported image, if it was saved with obstructions
		2. If available, a street view image should be displayed to aid the user with height estimations.
		3. Image location can be zoomed in or out to specify the desired area for the user.
		4. Image can be moved to center the image or move it to a more ideal position in the window for analysis
		5. Image can be labeled with solar modules using a location marker, with the user prompted to input heights from ground
		6. Image can be labeled with trees using Google Draw’s Circle Tool, with the user able to input heights from ground
		7. Image can be labeled with obstructions using Google Draw’s Polygon Tool, with the user able to input heights from ground
	5. Shade Percentage Calculation
		1. Calculate Sun Access through information from 1.4.5 to 1.4.7 and a ray casting algorithm
			1. Done through shooting a ray toward the sun every three minutes of every day, of every month for a year.
			2. If ray collides with obstruction, we say there is shade
			3. If ray doesn’t collide with obstruction, we assume there is sun access
		2. Output the average percentage of sun access for all solar modules on a per month basis, as well as an annual percentage.
2. Technical Tools
	1. Google App Engine Integration
		1. User Accounts handled through Google Accounts
		2. User Project Database accessed using a user key unique to
		3. Front-end created through HTML/CSS files
		4. Project system and calculation algorithm created and managed through python files
	2. Google Maps Integration
		1. Importation of an address, and grab image from that location, using average format (123 N. Main Ave. Cityville, ST 00000)
		2. Ability to view a street view, if available for a location
	3. Shade Percentage Calculation Tool and Image Drawing
		1. Solar Module Marker
			1. User will specify location of solar modules, as well as their height from the ground.
			2. Implemented through Google Map’s waypoint
		2. Tree Marker
			1. User will specify location of solar modules, as well as their height from the ground.
			2. Implemented through Google Draw’s circle tool
		3. Obstruction Marker
			1. User can specify locations of other obstructions in image (other buildings, large objects, etc.) along with their relative height to the solar modules.
			2. Implemented through Google Draw’s polygon tool.
		4. Calculation of Percentage
			1. Algorithm will take the heights of the solar modules, as well as the heights of the trees and obstructions to calculate the percentage shade for that location over the course of a year.
			2. Algorithm should calculate percentage up to 3% accuracy to SunEye Devices.

## Environmental Requirements

Our environmental requirements were based on how we believed the system would be used. Since solar employees aren’t always at an office, we believed the system should be web based, and support for smaller devices such as a tablet or phone, so they have the ability to use the application wherever they might be. In addition, the languages and APIs we used are also explained with reasoning behind the choice, such as choosing Google App Engine rather than creating our own user system and our own server.

1. The system must be web based.
	1. Google App Engine will allow us to handle user accounts through google accounts, making it more secure for the user.
	2. Allows for the user to access from anywhere there is internet access
2. The system must be mobile/tablet compatible
3. The system must include map integration.
	1. Google Maps is freely available, and can be used until the product is actually ready to be sold, at which point an enterprise license will be necessary.
4. The system must work on all modern web browsers.
5. The system will use the following languages and APIs
	1. The Google Maps API is officially available for JavaScript, and as JavaScript is available on such a wide variety of platforms, this will be our language of choice.
	2. As this is a web-based product, HTML and CSS are necessary for the creation of a front end that allows for the drawing on the Google Map, as well as an intuitive user interface to select and access projects.
	3. Google App Engine will allow for simple, secure user connectivity through Google Accounts, and integration with a database system that links with individual user keys. It also allows for easy connectivity between the HTML/CSS with the python files relating to the project system and calculation algorithm.
	4. AJAX was used in order to send data to the server for saving and loading projects, as well as sending information to the calculation algorithm. It’s also used when retrieving a specific user’s list of projects.

## Non-functional Requirements

There were two major non-functional requirements that needed to be noted by this project, and were given to us directly from Seth of Rooftop Solar. These relied solely on ensuring the initial shade estimate is very close to results outputted by the SunEye, and are accurate enough to be given to a potential client with confidence.

1. The algorithm for predicting the shade on solar modules shall produce accurate results within 3%, given the location and obstruction parameters.
	1. This margin of error is important to minimize because the calculation needs to be a suitable replacement of on-site inspections.
	2. The company wants to make accurate assessments for customers in determining whether they will have solar modules installed.
2. The inputting interface shall have a high usability so that it is easily comprehended by all users.
	1. Inputting details about on-site obstructions such as structure boundaries and trees (with their heights from the ground) is a critical step in reaching a calculated result, so the process needs to be uncomplicated.
	2. The algorithm used to calculate the result will take into consideration the smallest differences so the meaning intended behind the user’s input needs to be unambiguous.

# Architecture

Throughout development our architecture mainly went through very minor changes, but retained the same idea throughout. Our implementation relies on a front end website communicating with the Google App Engine, which contains components related to the user system, project management system and the shade calculator.



# Prescriptive Architecture

Our systems high-level architecture is a simple client-server model. The client side is made up of a web front-end. It serves the purpose of providing a graphical user interface, which allows a user to access all of the functionality of the system. All communication is done through http, with ajax where asynchronous interaction with the server makes sense.

The server side is made up of three main components: user management, project management and the shade calculator. The system also has a database, with tables for user and project data.

The user management component serves the purpose of account creations and allowing a user to log in and out. It receives a request to create an account, log in or log out, and then inserts or retrieves data from the database to complete the request.

The project management component is responsible for creating, saving, loading, listing, and deleting projects for a particular user. It receives an http request to create, save, load, list, or delete a project, and then inserts, retrieves or deletes data from the database to complete the request.

The shade calculator component is responsible for calculating the sun access available for a particular project. It receives an http request with a JSON object describing the different objects that were drawn on the map, along with their positions relative to the center of the map. This data is then run through the shade calculator algorithm (described in detail in our design document). The results are then put into another JSON object and returned to the client for display.

# Descriptive Architecture

Most of our implementation followed the prescriptive architecture plan very closely. The only difference being the user system. We intended to build our own user management system from scratch, but we later opted to use Google Accounts to handle users, as this functionality is already nicely integrated into the Google App Engine, and reduces the chance of security issues. This greatly simplified the project and allowed us to focus more on the design and implementation of the shade calculator.

### Implementation Details

The web front-end consists of HTML, CSS and JavaScript. The implementation exists in the ‘template-files’ folder. A python script, pyplate.py, is a small template engine that will generate the static HTML files, into a folder called ‘static’. These are the files that are actually served on the server.

On the server side, we used python with the webapp2 framework.

The user management component is a single python module ‘user.py’. It contains a single class that handles an http request for the user status. The api is ‘/api/user/status’.

The project management component is primarily in the ‘project.py’ module. It contains classes for saving, loading, deleting, and listing projects. The api calls are ‘/api/project/save’, ‘/api/project/load’, ‘/api/project/delete’ and ‘/api/project/list’. Another class for creating the project is in the ‘postProject.py’ module. It is separate because it’s api is based on the standard HTML post behavior.

Lastly the shade calculator component exists in the ‘calculate.py’ and ‘sunangles.py’ modules. The api is ‘/api/calculate’.

# Testing

We performed various types of testing including unit, integration, usability, and acceptance testing. For unit testing, we tested the python modules that make up our project. In integration testing, we made sure Google Maps, Google account login worked properly with our website, and managing projects worked well, we made sure the website was easy to use in usability testing, and we met with our sponsor to make sure we passed acceptance testing.

For unit testing, individual python modules were tested, specifically those pieces that contribute to the shade calculation algorithm where testing was further broken down along the process of determining sun access results. The first portion of testing this process was on the algorithm’s preparation work of the data it is given. The data comes from the Google maps interface which generates it as headings and distances from some point. The algorithm takes these headings and distances to generate x and y coordinates. Comparing these generated coordinates of the different objects represented by the data against the collection of relative object positions within the Google maps interface will show whether the preparation work was successful in its translation while reflecting the same position layout of the objects as intended. The results of these tests showed the accuracy of the algorithm’s preparation work and can be further examined within the Testing document.

For integration testing, we did various types of tests to make sure Google’s applications integrated well with our website and that our database could modify the data stored based on user commands. When testing Google Maps, we asked Rooftop Solar to provide us with a list of locations that they have previously visited. We then entered these addresses into the address bar to make sure that these locations would show up on the map. On our own, we also would feed the text box made-up data. All locations that Rooftop Solar provided us came up on Google Maps and if the address could not be resolved from our own testing, we made sure an error notification would pop up to let the user know that it could not find the address.

For making sure Google account would log-in, we needed to simply make sure that it worked for all types of Google accounts and that it recognized the login. We tested this feature with various normal Google accounts and special Google accounts such as NAU student emails. We did not have any problems with getting to the login page and having the website recognize the login.

Lastly, making sure that users can create, edit, and access projects is important. Our team made our own tests in putting various names and locations to make sure that it would appear, made edits to the projects, exited the site, reloaded the site to make sure it was saved, and made sure deleted projects would not reappear. We also asked various people including classmates and Seth Holland to look at our site to test this feature while performing usability testing. A bug was found in entering special characters in the name field, but it was quickly fixed.

Usability testing included having those that accepted to perform the test go through a worksheet we prepared and rate their experience while performing specified tasks. We were told there could be improvements made to the user interface to be more specific on the account creation. We also showed the website to some of the Rooftop Solar employees. They had various questions which led to changes in the text for our step-through instructions and they pointed out errors in our algorithm which we fixed by the final product.

Lastly, our acceptance testing validated by speaking to our sponsor Seth Holland about our system. Overall, he was very pleased as we exceeded his expectations. When we initially met with him, he expected us to integrate one solar module and one obstruction, but we have it working with multiple solar panels and multiple obstructions.

# Future Work

There are many improvements that we can make on our project. Since we only had a semester to work on implementation, there were many things that we did not get to. Some of these features included using a polygon tool for solar module placement, in-depth research on the possibility to derive heights of trees and obstructions from Google Maps, accounting for various tree shapes, better drawing tools, and creating a sort of authentication system to make sure that only people who are approved by Rooftop Solar would have access to the site.

Currently, we are using points to represent solar panels. However, solar panels have a surface area greater than a point and are usually placed with other solar panels to cover an area. In the future, we would like to implement the use of a polygon tool to cover an area and calculate the shade percentage on the covered area. In general, we would like to implement better drawing tools for the program. The system also only accounts for cone-shaped trees, so this is something we would like to change so that the shade analysis could be more accurate with different tree shapes.

Our system requires user input for obstructions and trees. We did not have the time to look too much into it, but it would be convenient to give the user a rough or exact measurement of certain obstructions given relative heights of objects on the street view. This would eliminate the need to go on site to measure obstructions so that estimates could be completed completely online.

Lastly, Rooftop Solar has expressed interest in licensing this site to other solar companies. In that case, only people with approval from Rooftop Solar should be allowed to access the site. Right now, anyone with a Google account can login to the site, so we would need to create a way for Rooftop Solar to give permissions to certain Google accounts once receiving payment. We have not received any negative feedback from our sponsor, so we believe that we fulfilled our acceptance testing.

# Conclusion

Currently in the solar industry, there is a considerable inefficiency in the amount of work is being done on solar estimations, as workers are being sent out to countless rooftops of potential contracts to make initial shade estimations, taking a good amount of company time and resources, with only a low percentage of these clients actually making a contract to buy and install solar paneling on their building.

Our web application we created has the ability to make accurate initial shade estimations instantly, using any internet connected laptop, tablet, or phone. This avoids the need to send people on site, while also getting sun access information back to the client sooner, allowing more focus on more potential and interested clients, rather than going onto the roof at every location.

The key benefits of this approach are:

* Significant improvement on time and resources for an initial sun access estimate
* Saves Rooftop Solar money for initial sun access estimates
* Plans to license the application can improve this issue for all solar companies, as well as create a new revenue stream for Rooftop Solar.
* With improvements, this application could replace the SunEye

By creating this project using a simple yet professional front-end website, a secure user and project database, and an accurate sun access calculator, this project will be a viable alternative for any company authorized to use it to make accurate solar shade estimates without wasting potentially important and needed company time and resources, and making their contracts with customers much more efficiently.

# Glossary

Ray Casting - The process of shooting a ray toward the sun, and checking if that ray collides with an obstruction or not. If it does, there is shade at that location. If not, then there is sun access.

Sun Access - The percentage of sunlight that hits a particular location.

SunEye - A physical device used by solar workers that takes a fish-eye lens photo upward to capture any obstructions that would affect the location’s sun access. It then outputs a percentage that states the sun access per month, or annually.