



Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: SolarUpp

Final Report

Fırat Ege Akın, Ömer Fatih Çelik, Alp Ertürk, Burak Korkmaz

Supervisor: Selim Aksoy

Jury Members: Uğur Gündükbay and Abdullah Ercüment Çiçek

Final Report
May 27, 2020

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1. Introduction	5
2. Requirements Details	6
2.1. User Specific Requirements	6
2.2. Data Sources	7
2.3. Non-functional Requirements	7
2.3.1. Reliability	7
2.3.2. Performance	7
2.3.3. Extendability	8
2.3.4. Scalability	8
2.3.5. Usability	8
3. Final Architecture and Design Details	9
3.1. Presentation Tier Subsystem	11
3.1.1. Web View	11
3.1.2. Web Controller	11
3.1.3. Model	12
3.2. Application Tier Subsystem	12
3.2.1. Application Management	13
3.2.2. Feasibility Study	13
3.2.3. Maintenance	13
3.3. Data Tier Subsystem	14
3.3.1. API Management	14
3.3.2. Plan Database Management	14
3.3.3. Inverter Database Management	15
4. Development and Implementation Details	15
4.1. Development Tools and Frameworks	15
4.1.1. Javascript	15
4.1.2. React	15
4.1.3. Node JS	16
4.1.4. Git	16
4.1.5. GitHub	16
4.1.6. Firebase	17
4.1.7. Firestore	17
4.1.8. CesiumJS	17
4.1.9. Resium	17
4.1.10. Image JS	18
4.1.11. PVGIS	18

4.1.12. Google Maps	18
4.1.13. React-Geocode	18
4.1.14. Prime React	19
4.1.15. Postman	19
4.1.16. Express	19
4.1.17. Axios	19
4.1.18. Node Geometry Library	20
4.1.19. Redux	20
4.1.20. Material UI	20
4.1.21. Weather API	20
4.2 Implementation Details	21
4.2.1. Javascript	21
4.2.2. React	21
4.2.3. Node JS	21
4.2.4. Git	21
4.2.5. GitHub	22
4.2.6. Firebase	22
4.2.7. Firestore	22
4.2.8. CesiumJS	23
4.2.9. Resium	23
4.2.10. Image JS	23
4.2.11. PVGIS	23
4.2.12. Google Maps	24
4.2.13. React-Geocode	24
4.2.14. Prime React	24
4.2.15. Postman	25
4.2.16. Express	25
4.2.17. Axios	25
4.2.18. Node Geometry Library	25
4.2.19. Redux	26
4.2.20. Material UI	26
4.2.21. Weather API	26
4.3. Calculation of Solar Estimates	26
5. Testing Details	28
5.1. API Testing	28
5.2 Test Driven Development	28
5.3 Unit Testing	29
6. Maintenance Plan and Details	29
6.1 Maintaining Third Party Services	29

6.2 Maintaining Real-Time Tracking of Solar Panels	29
6.3 Maintaining Performance	30
7. Other Project Elements	31
7.1 Consideration of Various Factors	31
7.2. Ethics and Professional Responsibilities	32
7.3. Judgements and Impacts to Various Contexts	33
7.4 Teamwork and Peer Contribution	35
7.5 Project Plan Observed and Objectives Met	37
7.6 New Knowledge Acquired and Learning Strategies Used	39
8. Conclusion and Future Work	40
9. Glossary	41
10. Appendix	42
Appendix A - User Manual	42
Appendix B - Class Diagram	53
11. References	54

1. Introduction

Each day our world is becoming polluted due to the usage of fossil fuels as an energy source. World Wide Fund for Nature, also known as WWF specifies fossil fuels as the biggest cause of climate change. Average global temperatures have risen almost 1°C since the industrial revolution [1]. As long as passing to renewable energy sources will not come true, levels of carbon dioxide and heat-trapping greenhouse gases in the atmosphere are going to increase. Renewable energy sources ensure reliable, sustainable, environmentalist and cost-efficient solutions rather than fossil fuels. Communities and governments have an increasing tendency to escape the impacts of fossil fuels and to build a livable world for the future.

Renewable Technologies are considered as clean sources of energy. Their usage will drop the negative effects of fossil fuels on social needs and future economics. The sun is counted as one of the biggest energy sources. It has the capability to provide more energy than we need to power everything in the world. The sun generates energy from a process called nuclear fusion. The generated energy radiates out to space by solar radiation. Solar Energy technologies are used to convert the power from solar radiation. The productivity of Solar Energy Systems is not the same everywhere around the world. However, there is a solar map for every position, time, location, and temperature. Therefore our thought as a group was “Why don’t we make a feasibility study on Solar Energy Systems to keep our world cleaner”. Our project SolarUpp will be based on energy production and efficiency on the rooftops of buildings to increase productivity. Moreover, it proposes tracking built solar panel systems with a cloud-based system. Information such as energy production, profit estimation, statistics, etc. will be tracked with this system. We are planning to make a notification system to inform users of emergency situations, and needs for care. Briefly, our project won’t be single-use software only. We are offering software as a service.

2. Requirements Details

2.1. User Specific Requirements

- Users able to register to our application by providing their personal email addresses or company email addresses.
- There will be 2 main service type within the system. First of this is feasibility estimating of solar panels. In other words, this is a calculation of the amount of solar energy production that the given rooftop is capable of. Second main service of this project is maintenance alarm system for solar panels. For now every verified user is able to see the feasibility estimation of solar panels and is able to use maintenance alarm system but for future work, maintenance alarm system service might be only for premium users.
- Users should be able to notified by maintenance notification system such as email from the web application itself.
- All users should be able to compare different solar panels and inverters to find optimum solution on their rooftops in terms of cost, size and efficiency.
- All users should be able to enter their addresses and find these addresses in Google Maps. They also should be able to see their buildings' picture from the top.
- Users checking feasibility of solar panels on a building should be able to see how the expected cost and electricity generating capacity are calculated. The variables used in the calculation should be presented clearly to the user.
- In Solar Panel systems, if the amount of solar energy produced is larger than the electricity needed, then solar panel users are able to sell the excessive solar energy to the government. Therefore users are able to see the expected profit if his expected solar energy production that is calculated by SolarUpp is larger than needed electricity.
- If a user has multiple buildings, he is able to compare the feasibility of Solar Panels to each of the buildings with their prices and estimated profit.
- If user cannot find a specific parameter that is for calculating feasibility and cost of the solar panels, he is able to enter all of the parameters manually.

- Each user is able to see the tutorial of SolarUpp and learn how to use the system.

2.2. Data Sources

- Archive of all generated electricity and their estimated electricity generation from tracked solar panels will be held in the database.
- Information of registered users such as address, email, phone number will be stored.
- Properties like size, price, efficiency, mass and energy production for different solar panels and inverters will be stored.
- The data about generated solar energy amount of verified users which is gathered from their inverter will be stored for distinct time intervals in the database.
- Solar maps and solar ratings of different regions will be provided to users before the feasibility study.

2.3. Non-functional Requirements

2.3.1. Reliability

- Users with installed solar panels should be notified if there is a problem in their solar panels, or the system. It should not take more than a week for anyone to learn about a malfunction in their solar panels.
- The values accumulated from the APIs should be accurate enough for expected cost and electric generating capacity.

2.3.2. Performance

- The application should run smoothly while browsing the map or checking for potential solar panel locations on a roof. The FPS rate should be 30 at minimum for the most part.

2.3.3. Extendability

- In case of the application reaching a critical level of user base, the system should be able to make great use of the gathered data.

2.3.4. Scalability

- The system should be able to serve to several different users at the same time.
- The system should be usable wherever used APIs are supported.

2.3.5. Usability

- Users should be able to find their desired building in a short time.
- The interface of the webpage should be easy to understand for users. The features of the application should be presented clearly.
- There should be a tutorial on how to use the application.
- The notification system of the application should be convenient for the users. For example, there should be more than one way to notify users.
- A user who had installed a solar panel prior to using the application should be able to use the application to track their solar panels.

3. Final Architecture and Design Details

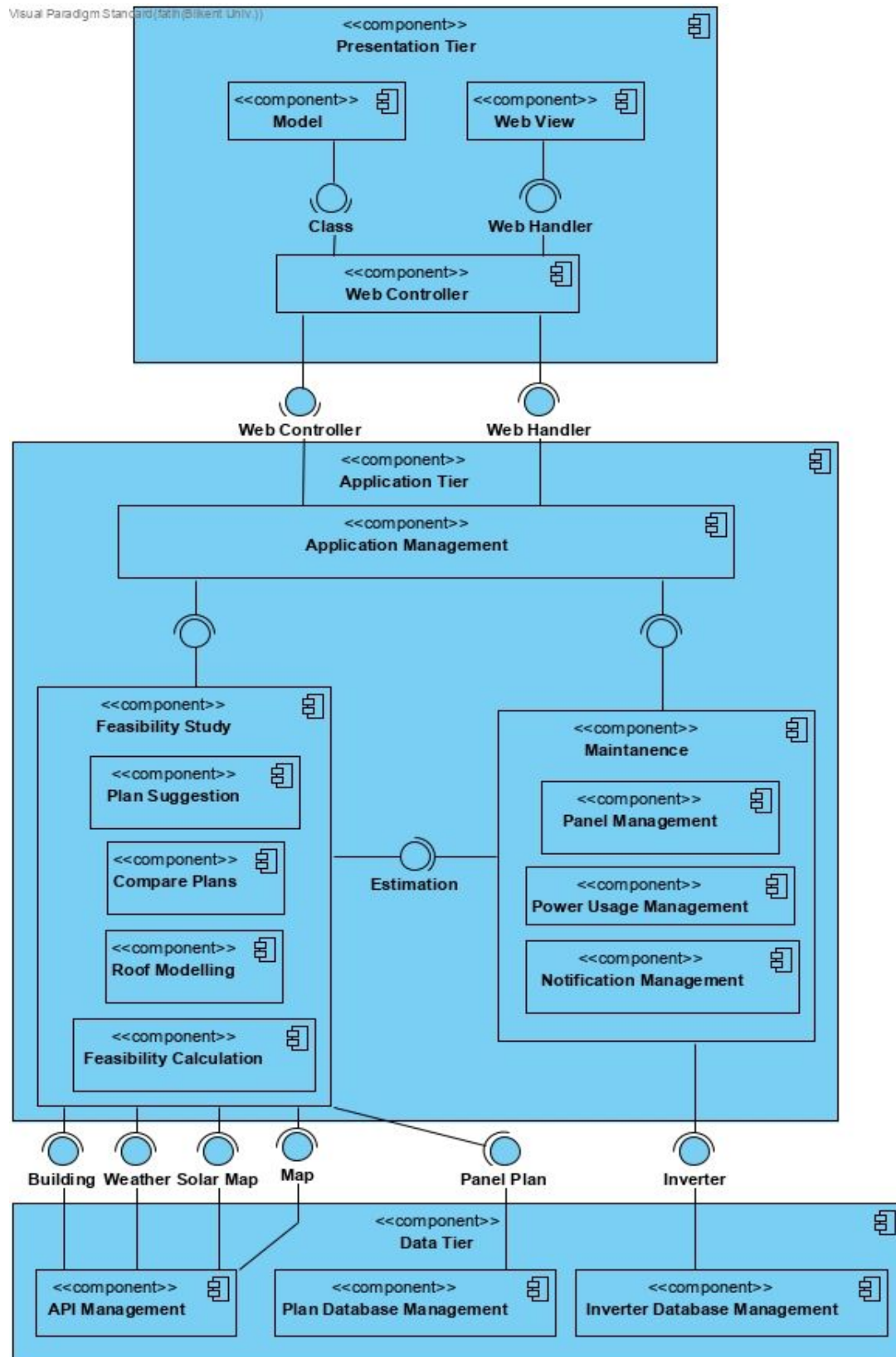


Figure 1: Subsystem decomposition

Subsystem Decomposition Description:

We will use the 3-Tier software architecture for SolarUpp. The tiers are Presentation Tier, Application Tier and Data Tier. At Data Tier, we have an API Management subsystem, Plan Database Subsystem, Inverter Subsystem and Panel Database Management. API management system manages fetching data from open source APIs that we use openWeatherMap. Plan Database Management component maintains all plans that are created from the SolarUpp system. The database Management component maintains all panels that are registered to SolarUpp and keep information about panels such as the amount of solar energy produced. Feasibility Study Component that is in the Application Tier is the main component and manages panel plan suggesting service, compare plan service that is for comparing solar panel plans of two buildings, roof modelling service that is for drawing panel plan to the user's building's rooftop and feasibility calculation that calculates the estimation of solar energy produced of panel plan. Maintenance Component manages the maintenance service that is for identifying failures of solar panels.

3.1. Presentation Tier Subsystem

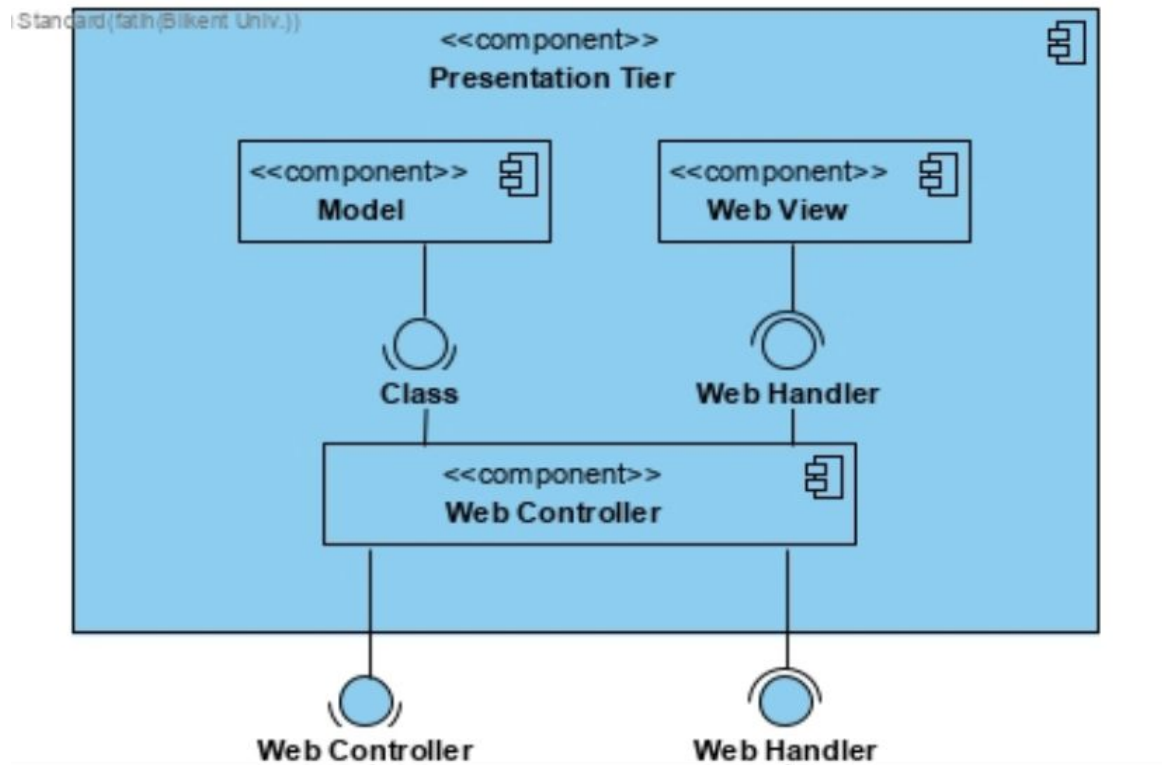


Figure 2: Presentation tier subsystem

Presentation Tier Subsystem provides the front-end of SolarUpp. It mainly controls the user interfaces.

3.1.1. Web View

Web View Component provides the interfaces that SolarUpp provides. If there is an update or change in the below tiers of the application, Web Controller handles and applies the changes to the Web View. CSS, Javascript, and HTML will be used to implement Web View.

3.1.2. Web Controller

Web Controller Component controls the Web View Component. It also provides communication between the application, data tier and web view. It reflects the

changes in data and application tier to the web view. React framework will be used to implement the Web Controller component.

3.1.3. Model

The model component is for MVC design pattern. We will represent some classes as models and this will reduce the complexity of code and duplicate code fragments.

3.2. Application Tier Subsystem

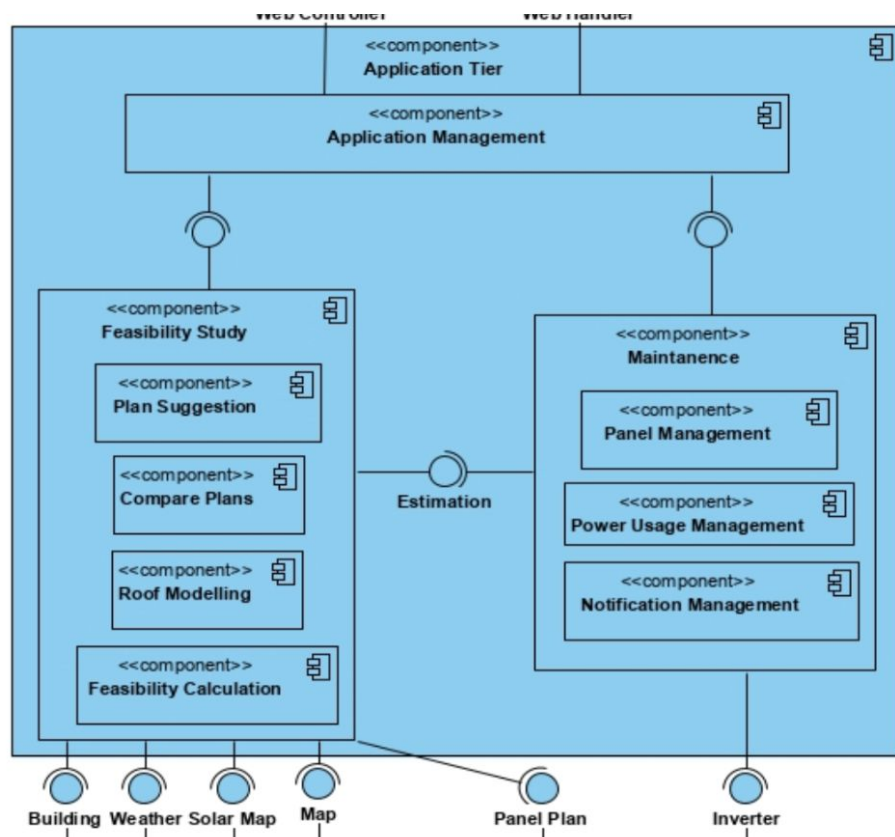


Figure 3: Application Tier Subsystem

The main services of SolarUpp are residing in the Application Tier Component. It also provides communication between the front-end and back-end of the SolarUpp system.

3.2.1. Application Management

Application Management component provides the communication between front-end and Application Tier.

3.2.2. Feasibility Study

Feasibility Study Component contains 4 subsystems in itself. These subsystems are the following: Plan Suggestion, Compare Plans, Roof Modelling, and Feasibility Calculation. Plan Suggestion component provides the Solar Plan Suggesting service that SolarUpp provides. This component contains machine learning models in itself and according to the machine learning model and datasets about solar panels, the Plan Suggestion component suggests the most suitable Solar Plan Model to the client's roof. The Compare Plan component basically calculates the difference between 2 solar plan models. The roof Modeling component provides the modeling of solar panels on the client's roof in terms of free spaces. Feasibility Calculation component provides all calculations such as feasibility score, estimation of solar energy produced, estimation of net profit.

3.2.3. Maintenance

The maintenance subsystem contains the Panel Management component, Power Usage Management, and Notification Management component. The notification component provides the alarm system that SolarUpp provides. If there is a defect in the solar panel, the Notification component notifies the client. Panel Management component provides an interface for real-life panels and it is collaboratively working with the Notification component. Notification Management component decides to notify the client with respect to information coming from the Panel Management component. Power Usage Management component stores information about the usage of solar energy. In total, the Maintenance component provides maintenance of solar panels of clients. It compares estimations that are coming from the Feasibility Study component with real-life calculations and decides whether there is a defect in

the performance of the solar panel. To implement the mentioned components, Firebase notification systems are used.

3.3. Data Tier Subsystem

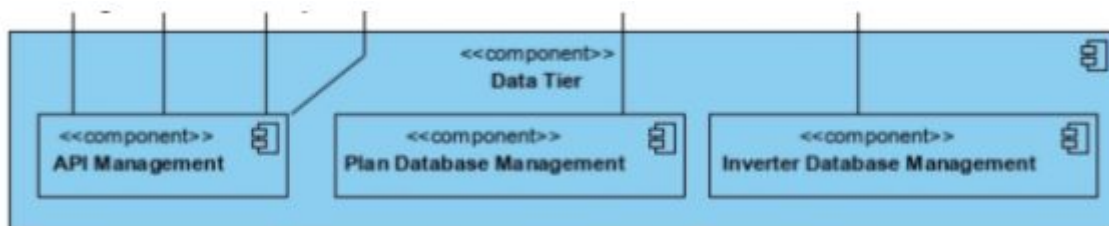


Figure 4: Data tier subsystem

The back-end services of SolarUp are provided in the Data Tier Subsystem. The API Management component will input various data from the world wide web to our system. Plan and Inverter Database Management components interactively manage the database side of our application.

3.3.1. API Management

This component handles the API requests for our application in order to gather data from different applications via their APIs. For more information on APIs, please refer to section 4 (Current software architecture).

3.3.2. Plan Database Management

The users of SolarUp are able to create solar panel implementation plans for their houses. All plans created whether incomplete or complete are stored in this component using Google Firebase. If a user decides to start on creating a plan and then decides to save it to complete the plan later, this component will handle the saving and loading requests for planning.

3.3.3. Inverter Database Management

This component handles all inverters connected to the SolarUpp. Inverter Database Management is like an interface for solar panels to our application. Every kind of data generated from inverters are stored in the Google Firebase by this component. When the Maintenance subsystem requests the stored information, This component checks the authentication and gives the raw data.

4. Development and Implementation Details

In this section, the development tools and frameworks used are showcased, then how they are utilised in SolarUpp is explained.

4.1. Development Tools and Frameworks

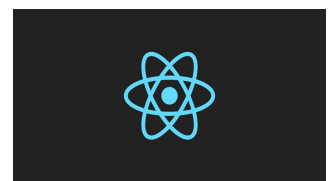
4.1.1. Javascript

JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles. [14]



4.1.2. React

React is an open source JavaScript library which is created by Facebook. Fundamental purpose of React is making web development easier with the help of component based design idea. React is used by developers to build interactive User Interface designs in a painless way. React requires Node JS development environment on the device in order to create a React



App and access related libraries. SolarUpp initialized as a React App and software structure is based on React methodologies. [18]

4.1.3. Node JS

Node JS is used as a runtime environment by web developers while producing dynamic web page content beside web browsers. Node JS is an open source, cross platform environment which works on most operating systems. SolarUpp runs on a npm server which is presented by Node JS for the development phase. In SolarUpp nodule managements are also done by npm (Node Package Manager) deals with required packages and dependencies in SolarUpp. [19]



4.1.4. Git

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.



Every Git directory on every computer is a full-fledged repository with complete history and full version-tracking abilities, independent of network access or a central server. Git is free and open-source software distributed under the terms of the GNU General Public License version 2. [23]

4.1.5. GitHub

GitHub is a development platform used by all kinds of developer groups, like open source and business. Users can host and review code, manage projects and build software alongside 40 million developers. [16]



4.1.6. Firebase

Firebase is a realtime cloud-hosted database system developed by Google. In firebase, data is stored as a JSON format and that data is synchronized in realtime in every client. Firebase lets users to use a nosql database system which is served realtime and cross-platform usability. [2]



4.1.7. Firestore

Firestore is a flexible NoSQL cloud database storage and data synchronization system between client and server. It offers users to store their data in a documents folder that contains field mapping to the values. Documents support certain data types, objects, strings, and images. Moreover, Firestore enables to deploy api functions to easily store, sync, and query data for mobile and web apps. [2]



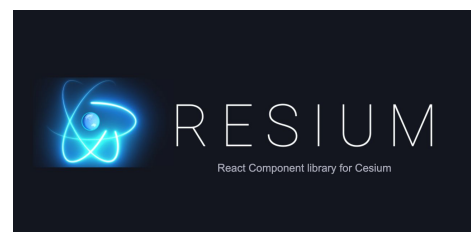
4.1.8. CesiumJS

CesiumJS is an open source JavaScript library for creating world-class 3D globes and maps with the best possible performance, precision, visual quality, and ease of use. This library is developed for Pure Javascript and offers good quality of imagery of the world and it has built-in find address service. CesiumJs is open source. It uses Bing Maps as imagery system, therefore Cesium requires developers to create a Bing Maps Api key. [4]



4.1.9. Resium

Resium uses Cesium as backbone. Resium is developed for React language. Its components are highly customizable. Resium is open source. It uses Bing Maps as imagery system, therefore Resium



requires developers to create a Bing Maps Api key. This library needs Cesium to be imported to the project since it uses Cesium as the backbone. [5]

4.1.10. Image JS

Image-js is an open-source image processing and manipulating library for Javascript. It is a full-featured library that can deal with simple image processing such as color leveling, edge detection and cropping etc. It has canny-edge-detector built-in for edge detection. [6]



4.1.11. PVGIS

PVGIS (Photovoltaic Geographical Information System) is a free online tool which is developed by European Union Science Hub which is accessible by everyone in order to have analysis for solar radiation data, photovoltaic performance and typical meteorological data around the world. PVGIS has its own data such as solar radiation maps, cloud mapping and detailed meteorological data. PVGIS is accessible by website as an online tool or it can be used by web requests. [20]

4.1.12. Google Maps

Google Maps Platform is a set of APIs and SDKs that are managed from the



Google Cloud Platform Console. It offers all google maps services such as geocoding, maps itself etc. For using Google Maps in React app, developers should create API access key. After that Google Maps component can be used and customized within the app.[8]

4.1.13. React-Geocode

React-Geocode is an open-source module for React which provides geocoding service for developers. Reverse-Geocoding is also available in React-Geocode.[22]

4.1.14. Prime React

PrimeReact is a collection of rich UI components for React. It is open-source and provides more than 70 components for React web applications. PrimeReact also provides some UI templates. These UI templates contain mostly PrimeReact components and developers are able to use these UI templates by only customizing and editing.[7]



4.1.15. Postman

Postman is a collaboration platform for API development. Postman's features simplify each step of building an API and streamline collaboration so you can create better APIs—faster.[17]



4.1.16. Express

Express is a Node.js web application framework which provides a robust middleware and HTTP utility methods to create APIs more quickly and efficiently. It is compatible with web and mobile applications. [3]



4.1.17. Axios

Axios is a promise based HTTP client for the browsers. It allows making XMLHttpRequest from the application, and HTTP requests from node.js.

Additionally it has a support for the Promise API which represents the eventual completion of an asynchronous operation. [10]



4.1.18. Node Geometry Library

Node Geometry Library provides helper functions for the computation of geometric data on the surface of the Earth. Its code is ported from Google Maps Android API. It can be used for calculations with respect to Google Maps Surface. For example, a drawn polygon area can be calculated by using the Node-Geometry Library.[9]

4.1.19. Redux

Redux is a stable container for JavaScript applications. It allows applications to behave consistent and run applications in different environments for client, server, and native side. It provides reducers to simplify code structure in an application [11]



4.1.20. Material UI

Material UI is a react component for faster and easier web development. With Material Design, users can build and design their applications with layouts, inputs, navigations, surfaces, feedback, data display, utils and lab components. It basically wraps css of your application [12].



4.1.21. Weather API

It is a weather API which is published by OpenWeatherMap that provides users the current weather data, hourly data, and 5 to 16 day forecast freely. Additionally it offers a professional option in order to fetch more robust weather and solar data [13].



4.2 Implementation Details

4.2.1. Javascript

SolarUpp software codebase consists of 88% Javascript, 11.1% CSS and 0.9% other types of assets. SolarUpp is powered by libraries built in Javascript like React and Node JS. Since React was used to power the front end, and Node JS was used to power the back end, hence, Javascript is embodied in the SolarUpp application.

4.2.2. React

SolarUpp created as a React App to increase ease of user interface design and have a more compatible development process by possibilities of React offered to developers in terms of dependencies, libraries and encapsulated component based design. Components in React have their own state which can be expressed as instant properties of these components when they are created and used. In SolarUpp, different components resemble different user interface design sections of our application. Different pages and small panels, dialog boxes created as components.

4.2.3. Node JS

During the development process to handle dependencies and nodule managements, we used npm. Node JS both provides a nodule package manager and a development server where we run our application. React requires Node JS since it needs some packages while creating the app. For SolarUpp we used Node JS as our JavaScript runtime environment, npm as package manager npm server as test server.

4.2.4. Git

Git was used in the implementation stage to collaborate on a single repository. Us group members concurrently worked on different functionalities of the application, sometimes even the same. With the help of git, we were able to continuously upgrade one master version of our application.

With the help of several features like history tracking, the project has a clean outline of what is taking place and where.

4.2.5. GitHub

GitHub was used to provide a front page of the SolarUpp project. GitHub has provided the project a public repository space for us group members to maintain a master codebase. Furthermore, GitHub was used to download the project reports to a public space. Although the development team was not big enough for some of its utilities to be useful (issue discussion etc.), GitHub was still a helpful tool to utilise git in the project.

4.2.6. Firebase

We use firebase as a base for our application. We bind database functions, authentications, database, store services of the Firebase under a project. We store documents in terms of collections which includes instances of documents. Our firebase project wraps all services and client side together. In the firebase console we can keep track of the changes in the database. It visualizes the result of the Postman test in the real world. Additionally, we bind database functions with the API of our firebase project. Using API services allow us to make simpler database connections on the client side.

4.2.7. Firestore

We store roof images, profile pictures of the users, authentication data in the firestore cloud services. The url links of the documents are connected in the necessary firebase database services. Using Firestore helps us to reduce the document size in the Firestore Database. In the free version of the Firebase Database services, registered document size is 1mb which is not enough for images. With the help of the cloud storage in the Firestore, we reduced the document size and kept documents efficiently.

4.2.8. CesiumJS

Google Maps does not use a globe view, Google Earth was providing globe view service but its library is not public anymore. Therefore since CesiumJS provides a high-quality globe-view, we used CesiumJS as the backbone of the Resium. CesiumJS is also used for setting Ion API. For using the Resium component the Ion api key should be setted.

4.2.9. Resium

Resium provides the whole functionality of Cesium for React language and using Resium was so useful for us since integrating Resium with our app was easy. Resium provides the globe view with multi imagery layers, therefore it is highly customizable. For example we added a polygon drawing feature to the Resium component and users are able to draw their roof by using the implemented drawing functionality on the Viewer component of Resium. We obtained coordinates of selected points by using the event listener of the Viewer component.

4.2.10. Image JS

Image-js library is used for image processing and manipulating. After getting the image of the roof of the user, we should make some processing in this image since we should detect the obstacles on the roof. For doing that we use canny-edge detection algorithm that is available in Image-js library. We also use crop functionality of Image-js to focus on only the roof of the user by cropping the unnecessary part of the image.

4.2.11. PVGIS

In SolarUpp, users are able to have feasibility studies for their buildings by sending proper requests to PVGIS interactive API. Users' information recorded such as latitude, longitude, free space on their roofs and this information is send to PVGIS as a HTTP GET request, where user information is added as parameters. Both in 'See Solar Plans' and 'Feasibility Study' pages where the user sends a request to

related API. PVGIS responds to SolarUpp as a JSON object which includes inputs (sent by user) meta data (explanation of sent data) and outputs which include energy production based on user's location such as daily, weekly, monthly and annually. By using this JSON object, SolarUpp calculates information such as estimated profit, cut carbon footprint amount and an estimated installation cost to present to the user as Solar Plans.

4.2.12. Google Maps

Google Maps component is mainly used for testing purposes. After finding the coordinates of the user's roof and calculating the results of the user's roof, we compared these results with the Google Maps results. While comparing with Google Maps results we found out some errors with our component and fixed them. [8]

4.2.13. React-Geocode

We only used Google Maps for testing purposes therefore in SolarUpp, there is no physical Google Maps component. Therefore we should be able to extract Geocoding functionality by using more specific library such as React-Geocode. The address information is essential for SolarUpp. Generating address string from coordinates is done by using the React-Geocode library. This library is used both at calculating the address of the computer and the address that the user enters while drawing his roof.

4.2.14. Prime React

PrimeReact was very useful for our project since it provides well-designed components for React. Therefore the user interface of SolarUpp is stable and very user-friendly. PrimeReact library decreased our workload on user-interface design. The chart components of PrimeReact are complex and high-quality, therefore these charts and graphs are used especially for SolarUpp's maintenance service and users are able to see their roof data within the PrimeReact charts.

4.2.15. Postman

Postman was used for API testing in our project. In the development of the application, it was useful to have a sophisticated tool designed to test our back end. We shared a request collection among ourselves to better integrate this powerful tool to our project. We were able to perform unit testing, functional testing and security testing.

4.2.16. Express

We used express js as a wrapper for our database connection, utils, and the database functions. We create a REST API with the help of express js. Deploying a REST API on the firebase lets you automatically run backend code in response to HTTPS requests.

4.2.17. Axios

We use axios to make the database connection with the client side of the application. Axios allows us to make API requests for database services, weatherServices, PVGIS, and authentication services. With the usage of the Axios, API requests such as GET, and POST are handled.

4.2.18. Node Geometry Library

While calculating free-space available on the user's roof, we implemented a Polygon area calculation function. After that we want to test whether our function's results are precise or not. Since we don't use a physical Google Maps component within our app, we should use a specific library that provides world surface calculations such as Node Geometry library. We calculated the polygon area which is basically the roof area by using Node Geometry Library. Node Geometry Library uses Google-Maps' geometry. Therefore our results become more precise after using this library.

4.2.19. Redux

By using Redux, we connect components and styles in the components within the applications. It helps to transfer the states within the pages. It wraps the user actions and data actions in a separate directory to simplify the overall structure. In redux, we used a type called reducer to transfer the state of the pages. Types for authentication, users, errors, loading ui, and errors are shared by that the reducers.

4.2.20. Material UI

We used Material UI design icons and labels for the login, signup, and profile page designs. It offers a simpler and futuristic design for components and simplifies the css theme process of the design of these pages.

4.2.21. Weather API

In the maintenance services, Weather API is used to calculate the weather estimation for the following 5 days. The Weather API offers a wider range of services with some charge. However, for the demo purposes we decided to go with the 5 day estimation.

4.3. Calculation of Solar Estimates

When the user finds his address on the map, he is now able to draw the roof constraints. When he clicks a red dot is generated on the map which indicates the constraints of the roof. When user clicks, the latitude and longitude are saved into coordinates array. These coordinates are calculated from the action listener of the map component. Also screen positions relative to that map component are stored in screenPositions array. If the user did not like the drawing of the plan he is able to remove the plan. When he clicks the remove plan button, screenPositions and coordinates array is resetted.

When the user finishes drawing the roof plan, a polygon is created on the map. This polygon is created by using the user's latitudes and longitudes. The area of the polygon is calculated for calculating the roof area. We used Google Maps geometry

for calculating the area of the roof. Total space is calculated in the Find Address Page and after that in the Feasibility Page, edge detection is done. After edge detection the white pixels demonstrate the obstacles on the roof. And the free space is calculated by extracting these white pixels from total space. Screen positions are used for taking the screenshot of the user's roof.

After users choose their building and free space on the roof is calculated, they are directed to the feasibility study page to get information about solar potential of their buildings. Users need to enter required information such as their roof material, roof angle, their monthly average electricity consumption in order to get a more precise feasibility report. Some other parameters required to have solar feasibility study such as latitude, longitude, peak power of featured system, foreseen system loss are sent to PVGIS API as an axios request. PVGIS is a free online tool which is developed by EU Science Hub [20]. PVGIS includes essential information for solar study such as solar radiation maps around the world, cloud mappings and also meteorological data. PVGIS API receives user specific related parameters which are entered by user or calculated by SolarUpp and do feasibility study by considering selected roof's global position to get correct solar radiation, cloud mapping and meteorological information to calculate estimated solar energy production. After calculations done by PVGIS, it sends a JSON file which includes daily, monthly and annual solar energy production of selected building. After receiving data about energy production, calculations are made to present a feasibility report which includes the user's estimated profit in the following twenty five years, cost analysis and amount of reduced carbon footprint. For maintenance service of SolarUpp, estimated solar energy production of existing systems are calculated by using PVGIS API. After a user gets his estimated production, he is able to upload his own production data of existing solar systems as '.csv' file to track performance of solar system and to see the difference between production of solar system and estimated production. Performance of solar system presented by graph between comparison of actual data and estimated data..

5. Testing Details

In this section the testing strategies used are discussed.

5.1. API Testing

APi testing in the SolarUpp project was done with the Postman application. For example, after implementing a back end function to our system, we continued forward with testing its behaviour by placing different kinds of requests.

We were able to test different kinds of use cases, like what would happen if a non user tried to use our services. We were also able to change the json file parameters from the Postman to further investigate the performing of our back end system.

This kind of testing allowed us to carry on with our development process knowing what our back end system is capable of accomplishing, hence, we would know if a problem was caused by a front end development or the back end development.

5.2 Test Driven Development

First of all, Test Driven Development is a software development process which depends on turning the requirements in the test cases. Then the code is improved to past the specifically designed test cases. In the development progress of the project, we go with the Test Driven Development to implement the specifications. For each development process we convert all necessary requirements into test cases. Initially all cases failed. Next, we implement the code just enough to pass the test cases and refactor the code.

The purpose of the Test Driven Development is to develop the optimal solution for the given specifications. That goal helps us to eliminate unnecessary code structures and minimize the software code of the project.

5.3 Unit Testing

We used White Box Testing method for unit testing for having feasibility study by sending requests. For unit testing we used JWebUnit to write test cases for our web application and used Chrome Coverage (one of Chrome development tools) to inspect coverage of our test cases for related functionality.

6. Maintenance Plan and Details

6.1 Maintaining Third Party Services

SolarUpp uses various services and libraries. Some of the services that SolarUpp uses are not free. These are Resium, Google Maps, Weather API, Firebase and Bing Maps. They all necessitate priced API keys to work. Up to this step of the project, we use the demo versions of these services since we did not get a lot of requests to our website in the development phase. Since we would get users to our system in the future, we should provide priced versions of these services because after a request threshold, these services become priced.

6.2 Maintaining Real-Time Tracking of Solar Panels

Due to the Covid-19, we could not obtain real-time solar panel data from companies that we consulted. Therefore we provided manual solar panel data for this phase. Since SolarUpp provides maintenance service, we should integrate the user's panel and inverter with SolarUpp. For doing that we thought to use SolarEdge's monitoring API. SolarEdge provides tracking real-time panel's data. It uses REST api as backbone, and the monitoring service can be obtained by SolarEdge API. But we observed that SolarEdge is also priced. SolarEdge also sells solar panels and the api key is given only if a user bought a solar panel from Solar Edge. Therefore we could not obtain API key from here but for future we plan to maintain real-time

tracking of solar panels by integrating SolarUpp with SolarEdge. Users only need to give the API key that is available when they bought the solar panel from SolarEdge. After that we will send API requests with the user's API key to SolarEdge and get real-time solar panel data.

6.3 Maintaining Performance

Since we use a lot of libraries and APIs, we have a lot of http requests, so scalability is essential for SolarUpp. While implementing the project we paid attention to request small objects from the backend. Therefore since the size of the transferred object is small, time-efficiency of frontend - backend communication is high. Also we paid attention to store only the significant data in our database. Therefore it can be stated that we provided storage-efficiency for our backend.

7. Other Project Elements

7.1 Consideration of Various Factors

7.1.1 Covid19 Pandemic

During the development process, we were also in contact with several companies which work on solar panel installation and Gazi Teknopark which includes the first solar lab in Turkey. Due to Covid19 pandemic after a while we were not able to meet with these contacts any longer. We were planning to get reliable and live data from these contacts in order to use it in 'Maintenance' part of SolarUpp where users can track their installed solar systems. Since, we can not get live data from these contacts we fetched data online to use in the Maintenance screen. We are able to fetch and inspect data but we would prefer more reliable and detailed data from our contacts.

7.1.2 Wide range of solar systems:

Solar systems components such as inverters and panels have a wide range of variations. Some solar panels can be used only building integrated where some of them are only free mount. Since some inverters have only data loggers and do not have Wifi connection, in SolarUpp we preferred to present up to date and compatible components with each other.

7.1.3 Meeting with professionals and research people:

In the early stages of the development process we met with companies such as Halk Enerji, AnkaSolar and research people in Gazi Teknopark to have domain research and understand regulations about solar energy in Turkey. These meetings both improved our understanding from technical and business perspectives. We had information about fundamental points about solar system installation, different tools around the world for solar analysis and possible rivals about both maintenance and feasibility tools. These meetings and talking with related people were extremely beneficial for our product.

7.1.4 Inconsistency about solar regulations in Turkey:

Laws and regulations about solar energy changes so frequently in Turkey like maximum capacity that people can have on their roof, or how people can sell their produced electricity. In SolarUpp, we designed our application more flexible than recent regulations because we believe in following years green energy should be maximized around the world.

Factor	Score	Effect
Covid19 Pandemic	3	Negative
Wide range of solar systems	1	Negative
Meeting with companies and research people	8	Positive
Inconsistency about solar regulations in Turkey	1	Negative

Table 1: Consideration of the various factors

7.2. Ethics and Professional Responsibilities

The professional and ethical issues related to SolarUpp discussed in terms of data retrieval and storing user specific information. Some amount of data required for our project is not stored as a database. In order to use these data such as properties of solar panels or inverters data should be copied to our application by hand which will not have a legal limitation because solar panel properties are shared publicly. Solar maps of different areas around the world are distributed without any restriction so it will not have a legal concern. Users' private information such as address, email or phone number will not be distributed or shared.

7.3. Judgements and Impacts to Various Contexts

In this section the informed judgments we made in our project that consider the impact of our solution in global, economic, environmental, and societal contexts are discussed.

Judgement Description: Making the software free to use	Impact Level (0-5)	Impact Description
Impact in Global Context	4	Users are free to test and use our software, there is nothing to lose for anyone.
Impact in Economic Context	5	Economically this judgement allows everyone to access free services.
Impact in Environmental Context	4	Environmentally a free tool like SolarUpp increases the green energy use.
Impact in Societal Context	3	Free to use softwares creates a paradigm that supports collaboration between people regardless of who they are.

Table 2: Consideration of making the software free to use

Judgement Description: Listing retail inverter and panel products	Impact Level (0-5)	Impact Description
Impact in Global Context	2	Since we can't list all of the retail products, in the global context this judgement only affects those who can access to the retailers of the listed products.
Impact in Economic Context	5	In the economical context this judgement creates a huge effect as

		the users get to know some of the retail prices of the products.
Impact in Environmental Context	3	In the environmental context this decision helps people choose the right products to set up a system for their needs, allowing them to use more efficient green energy.
Impact in Societal Context	1	In the societal context this decision may make some people think different of the listed retailers from some particular regions.

Table 3: Consideration of listing retail products

Judgement Description: Cut carbon footprint calculation	Impact Level (0-5)	Impact Description
Impact in Global Context	4	People from all around the globe can reduced amount of their carbon footprint by having solar systems.
Impact in Economic Context	2	In the economical context this judgement will promote using green energy sources.
Impact in Environmental Context	5	This is a very healthy judgement in environmental context as the carbon footprint directly affects the environment.
Impact in Societal Context	3	This judgement may cause people to see others with possible higher carbon footprint as less of a human.

Table 4: Consideration of carbon footprint calculation

Judgement Description: Allowing users to compare their systems between them	Impact Level (0-5)	Impact Description
Impact in Global Context	2	This judgement is not very effective for people all around the globe.
Impact in Economic Context	3	This judgement changes the peoples perspectives on different kinds of systems.
Impact in Environmental Context	4	This judgement changes peoples' impact on the environment in a positive way.
Impact in Societal Context	0	This judgement does not cause any societal impact.

Table 5: Consideration of system compare functionality

7.4 Teamwork and Peer Contribution

In terms of team work we decided to adapt the project plan which was defined on the analysis report(also you can refer to section 7.5). Projects parts such as database, client, functions, components, and API's are divided between the project members according to their knowledge.. During the process of the project we applied the analysis, design, development/implementation, and testing/reviving process as it was assigned to each member. Each member was mainly responsible for a seperate parts. Sadly, we all faced a pandemic crisis Covid-19 which forced us to work remotely. Before the Covid-19 situation, we had arranged meetings every monday in the Bilkent main library.

The time period before the pandemic crisis mainly included analysis, design and some implementation parts. The implementation and the development part is mainly handled remotely. We arrange zoom meetings and discord meetings during the process, and we publish the development processes in the projects GitHub page

[<https://github.com/egeakin/SolarUpp>]. In the following figure you can see the contribution of the project members in terms of commits and code changes.

Oct 13, 2019 – May 26, 2020

Contributions: Commits ▾

Contributions to master, excluding merge commits

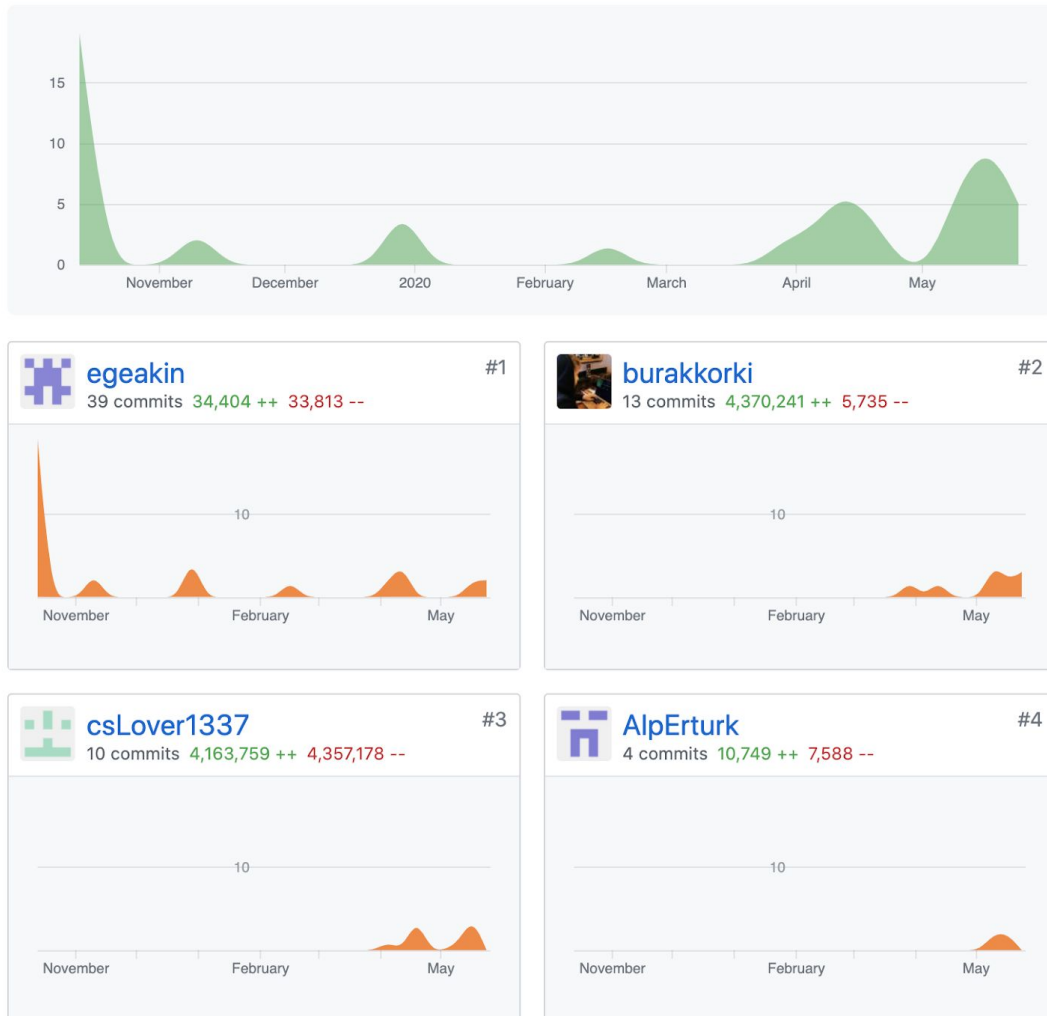


Figure 5: Team member commits over shared repository

In total, there are more than 90 commits during the implementation. Additionally we use two branches to make the necessary updates in terms of the git structure. You can observe the network in the project as follows.

Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.

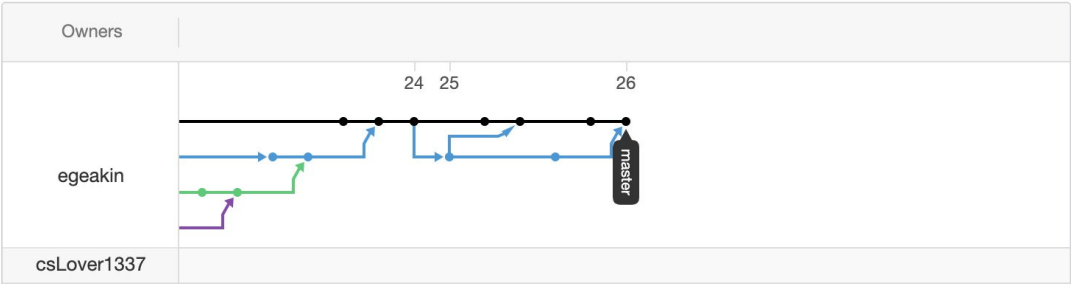


Figure 6: Source control chart

Overall, all of the team members follow the predefined requirements and attend all of the meetings during the process. Since we can arrange online meetings more than ever, we decrease the merging problems that might occur. We can say that we use git structure good enough to handle such a full stack project.

7.5 Project Plan Observed and Objectives Met

In the analysis report, we introduced the project plan in terms of the Gantt chart and its diagram. These diagrams basically wrap all the development progress of the project.

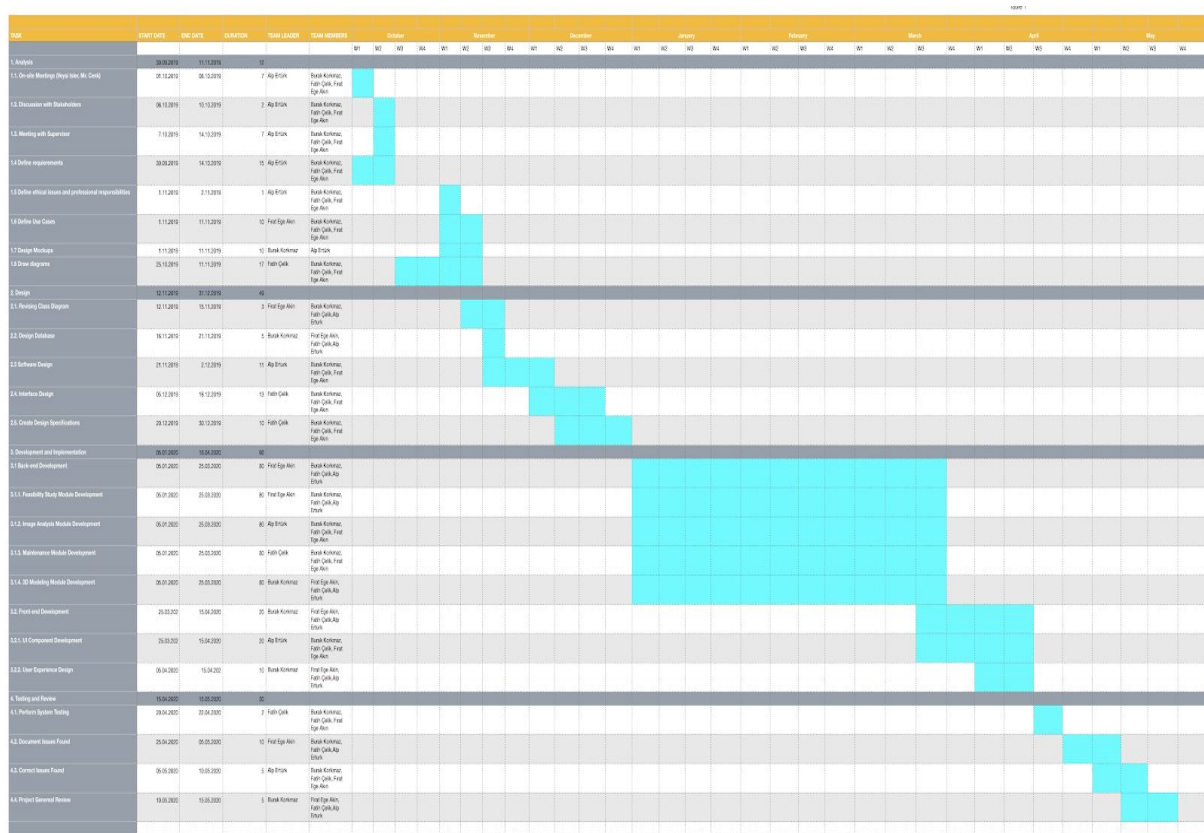


Figure 7: Gantt chart of the project lifetime

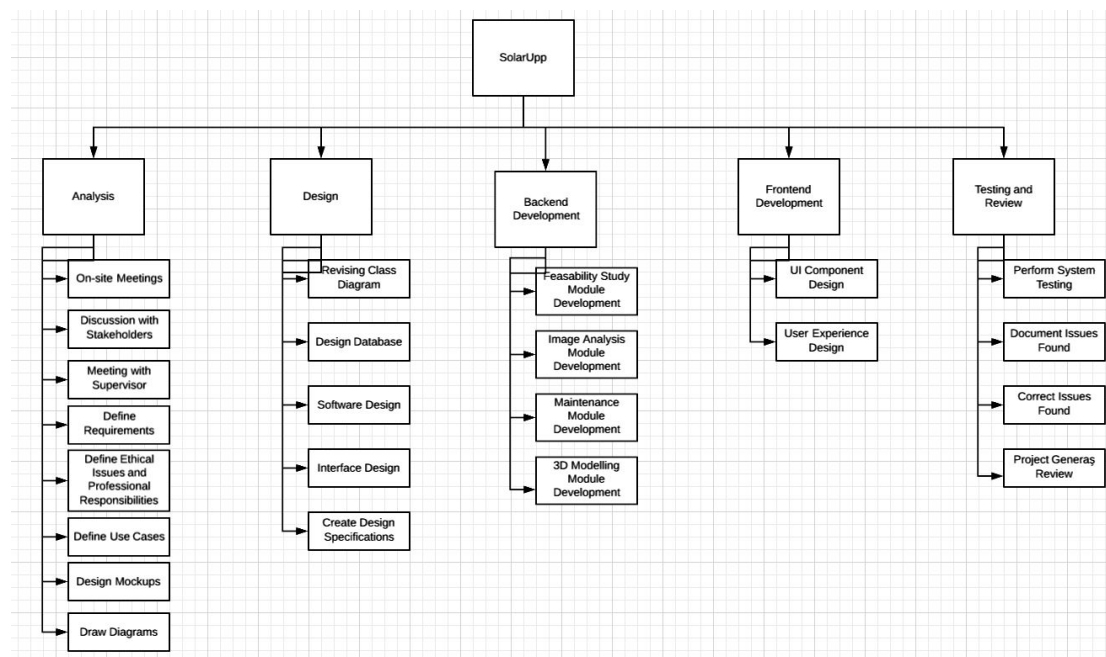


Figure 8: Grouping of the project tasks

During the implementation process, we are strict with the initial plan. There were some delays because of the remote working process, and all pandemic process. However we completed the requirements that we defined earlier successfully. We combine the development and testing process in terms of Test Development Environment to build the software with less error possible. Additionally, we tested all database functions by using Postman to ensure it captures all possible cases. In the end, we were successfully implement the overall mock ups that we designed at the beginning of the project.

7.6 New Knowledge Acquired and Learning Strategies Used

SolarUpp will be a web application developed on the React framework, which is JavaScript based. Our application structure will follow the client-server model; hence, we will need to learn database management with React as well as developing the necessary features our application requires with React and JavaScript. We will follow online tutorials for the most part. We will also examine other projects that are similar to ours done with the React framework.

We will also need to work with different kinds of APIs; for example, we will need to access weather and solar information in the run time of the application. We will have to learn interfaces of such APIs and how to implement their services in our application. We will review the documentation of these APIs and learn how to access their services using the React framework.

Since our project is about solar panel implementation, we will need to improve our problem domain knowledge drastically. We will learn the process of both how solar panels are implemented and how they are maintained. We will interview with the experts in this domain, by the means of either face to face or online interview. Fortunately, our innovation expert Veysi İşler has some domain experience; we are planning to have interviews with him about the proper steps to take in our project. Furthermore, we will also make our own analysis based on online literature.

Although it will not be feasible without a critical mass of users, we want to use machine learning in our application to analyze the data gathered from users. Our plan is to use a machine-learning algorithm to find similar buildings and pair those buildings with successful solar plans and buildings that need a solar plan. Three of us in the group are currently taking the introduction to the machine learning course,

and we plan to interview data scientists from Bilkent if we get to a critical mass of users.

8. Conclusion and Future Work

We wanted to create an application that would have a positive impact on the world without a doubt. Since it is a well known fact that the non-renewable energy sources are one of the biggest threats for the earth, we had decided that putting our time in a work that promotes renewable energy would be a time well spent.

We made a brief research over the utilization of solar energy in Turkey and we decided that creating a software that helps people to make the decision if to switch their energy source from grid to a solar system.

This project has been a tremendous opportunity for us to participate in various parts of a typical software engineering project. We were challenged in many different stages and had to overcome countless problems. Quickly finding creative solutions to the never ending problems list has made us more experienced in the solution domain of software engineering, while our path to understand the reality of the problem domain has made us appreciate the challenges of software engineering.

The significance of being collaborative is better understood by all of us, as well keeping a clean progress of work and codebase has been too. We first handedly experienced that while designing a perfect system is impossible, when the more is changed in the implementation stage, then the more complexity increases and the more time the development takes.

Some of the future works that can take place is to implement machine learning algorithms to serve more functionalities in the application. Some of these functionalities would require the application to expand to a critical size first, as without sizable data on hand the results wouldn't be useful. Some of the functionalities that can be implemented are finding similar systems in regards to the environmental conditions (weather, solar etc.) to compare, finding the best setup of solar panels in a roof without manual user event.

Other works that can be done is to change the way how some of the functionalities work as feedback is generated from the users.

9. Glossary

- **Rooftop:** The outer surface of a roof
- **Irradiance:** The density of radiation incident on a given surface usually expressed in watts per square centimeter or square meter
- **Camel Case:** Naming convention used in programming.
- **UML:** Standard used for modeling and visualization of components in software.
- **URL:** Uniform Resource Locator, a reference to a web resource, address.
- **API:** Application Programming Interface
- **Framework:** A platform for developing software applications.
- **REST API:** REpresentational State Transfer Application programming interface.
- **Gantt Chart:** A bar chart that illustrates a project schedule.
- **Canny Edge Detection:** An edge detection operator that uses algorithms such as gaussian blur, sobel operator, histograms etc to detect a wide range of edges in images.

10. Appendix

Appendix A - User Manual

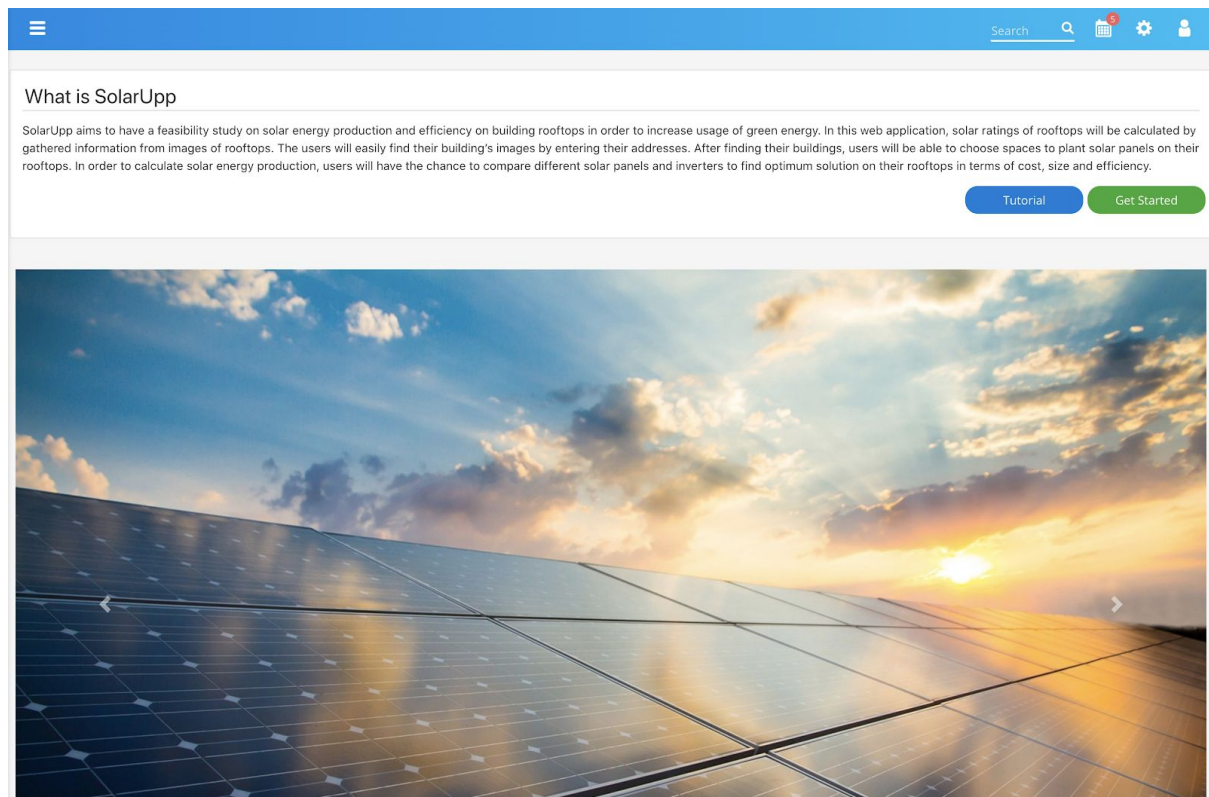


Figure 9: Start screen

This is the start screen of the SolarUpp, there is information about the project. Tutorial section introduces the application to the potential users. Get Started option directly load to screen to the login/signup page or find address section.

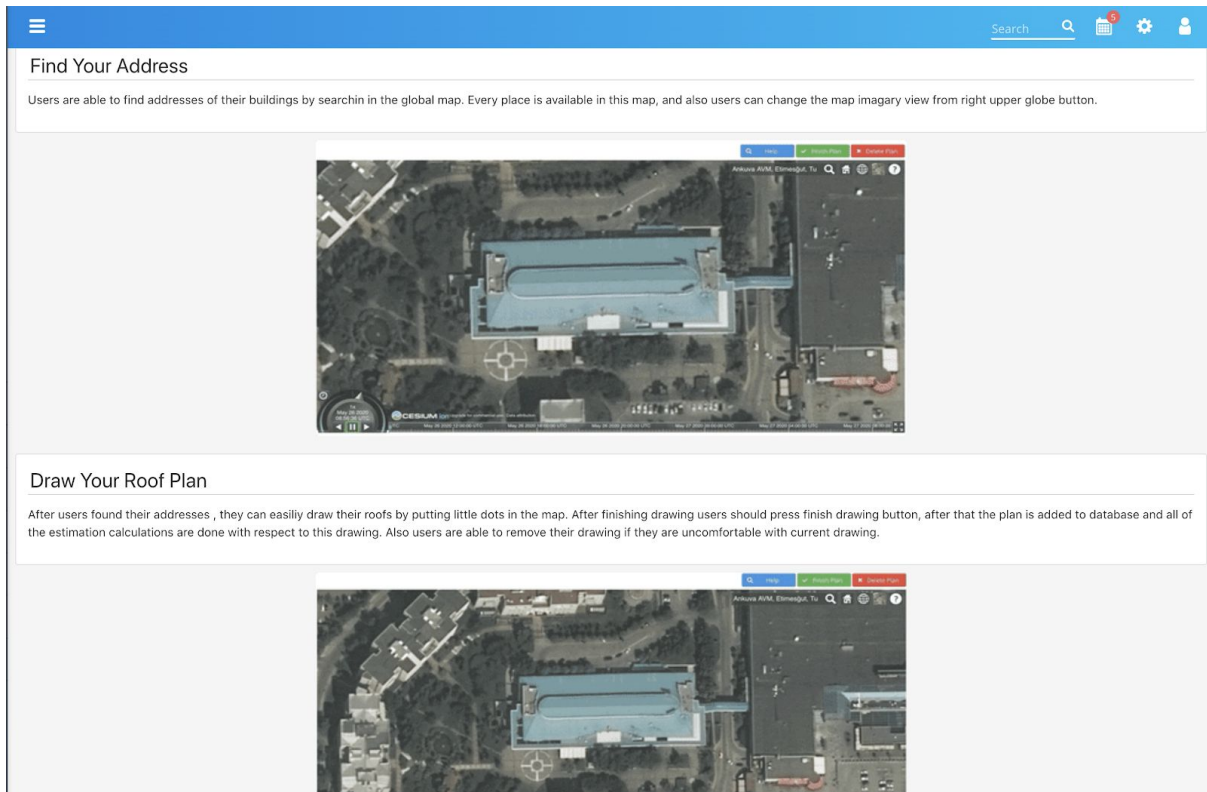


Figure 10: Tutorial screen

This is the first page of the tutorial section. It introduces how to find address and draw the roof according.

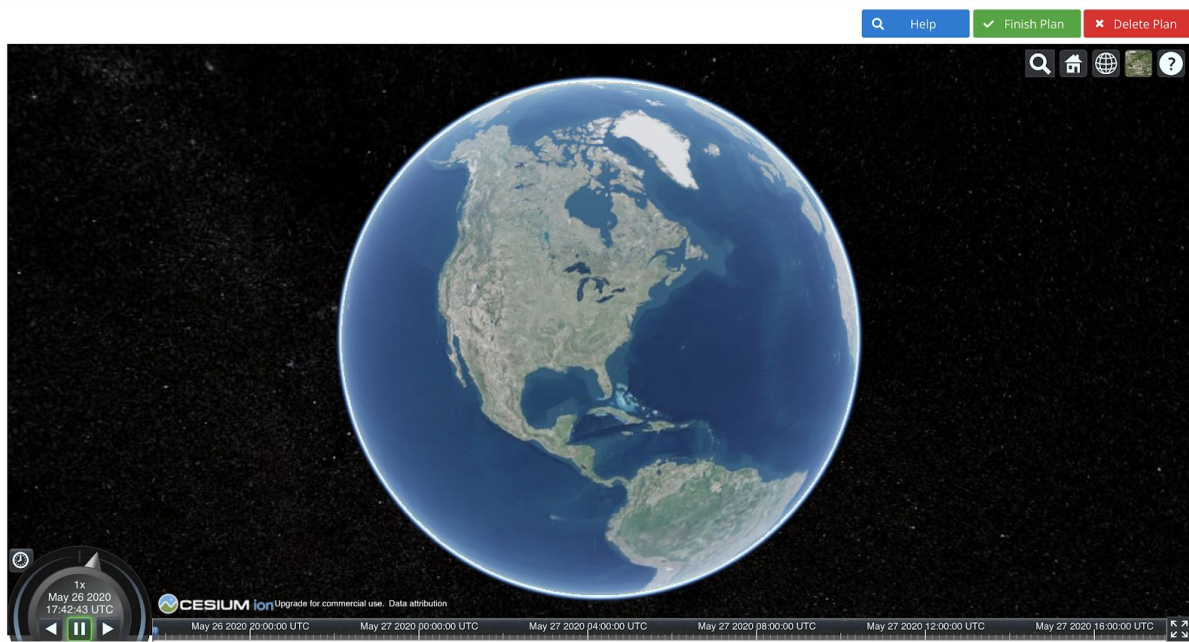


Figure 11: Find address simulator

There is a draw roof simulation which introduces how to draw a roof in the find address section.

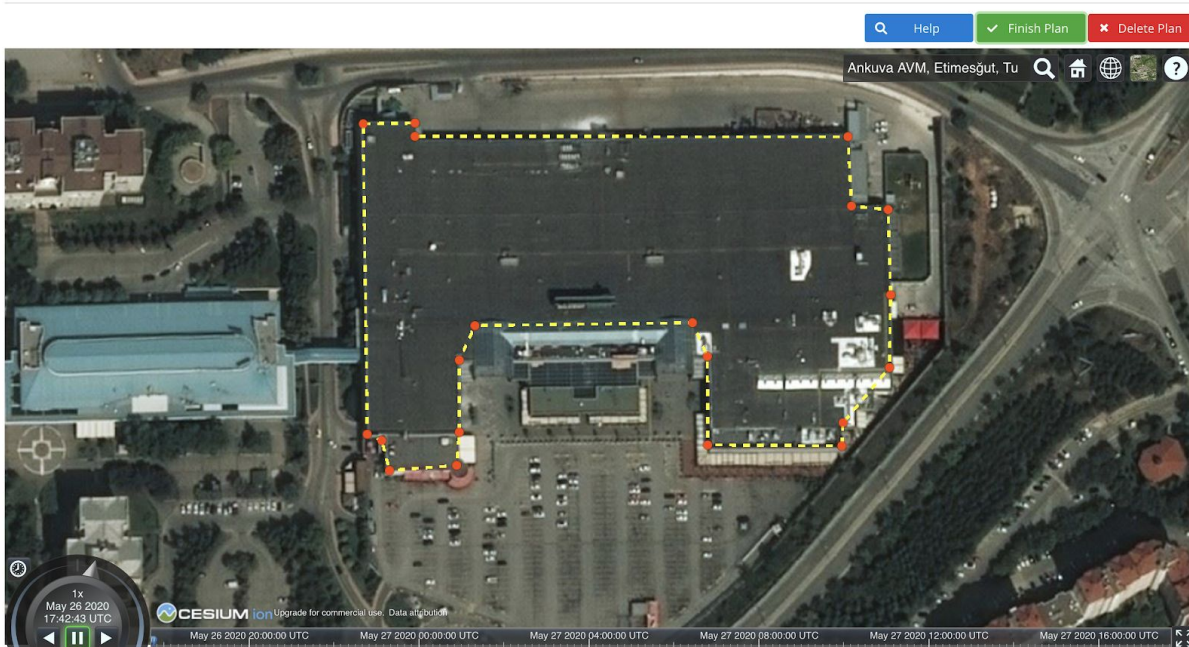
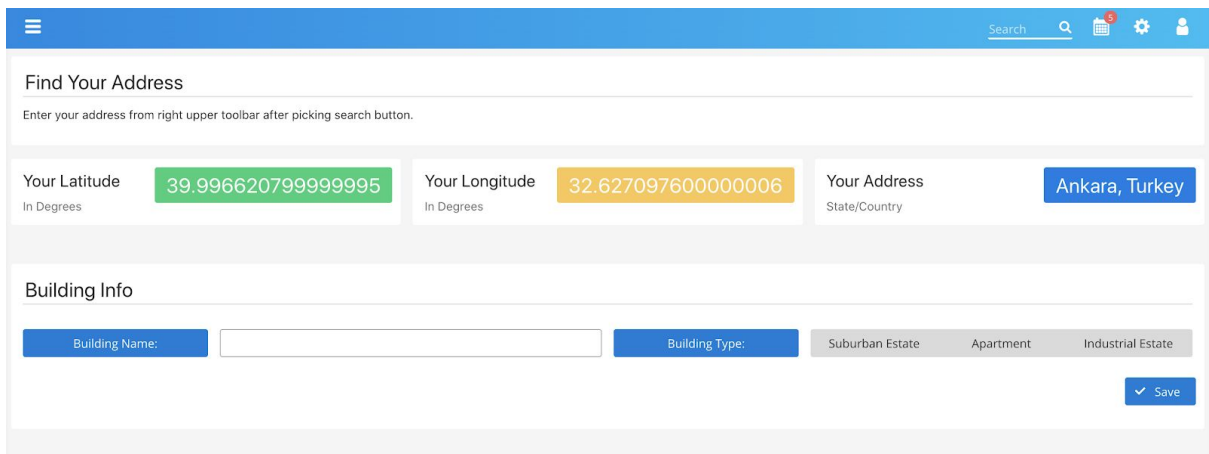


Figure 12: Find address simulator example

This is the final version of the draw roof simulation on the introduction page. Now users can able to draw their roofs as it was introduced in the introduction.



The 'Find Your Address' overlay form is located at the top of the application. It features a blue header bar with a search icon, a calendar icon with a red notification badge, a settings gear, and a user profile icon. Below the header, the form is titled 'Find Your Address' and includes a placeholder text: 'Enter your address from right upper toolbar after picking search button.' The form is divided into three main sections: 'Your Latitude' (displaying 39.996620799999995), 'Your Longitude' (displaying 32.627097600000006), and 'Your Address' (displaying Ankara, Turkey). Below these sections is a 'Building Info' section with a 'Building Name' input field, a 'Building Type' dropdown menu (with options: Suburban Estate, Apartment, Industrial Estate), and a 'Save' button.

Field	Value
Your Latitude (In Degrees)	39.996620799999995
Your Longitude (In Degrees)	32.627097600000006
Your Address (State/Country)	Ankara, Turkey

Building Info

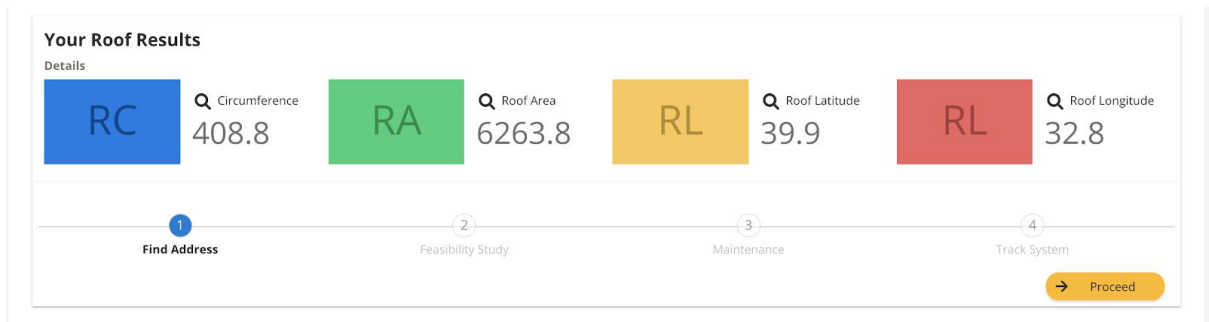
Building Name:

Building Type: Suburban Estate Apartment Industrial Estate

[Save](#)

Figure 13: Find address overlay

Here, users can see their computer's latitude, longitude and address. Also before starting a new solar plan, user enters the building info in this page.



The 'Your Roof Results' overlay displays the results of a roof analysis. It features a blue header bar with the title 'Your Roof Results' and a 'Details' section. The results are presented in four colored boxes: 'RC' (blue) for Circumference (408.8), 'RA' (green) for Roof Area (6263.8), 'RL' (yellow) for Roof Latitude (39.9), and 'RL' (red) for Roof Longitude (32.8). Below the results is a progress bar with four steps: 1. Find Address, 2. Feasibility Study, 3. Maintenance, and 4. Track System. A 'Proceed' button is located at the bottom right of the progress bar.

Field	Value
RC (Circumference)	408.8
RA (Roof Area)	6263.8
RL (Roof Latitude)	39.9
RL (Roof Longitude)	32.8

Progress Bar:

- 1 Find Address
- 2 Feasibility Study
- 3 Maintenance
- 4 Track System

[Proceed](#)

Figure 14: Find address overlay

After user draws his plan, the roof results are visible here. Users can see circumference, area, latitude and longitude of the roof. After that, by clicking proceed the user continues to his plan and goes to form feasibility study.

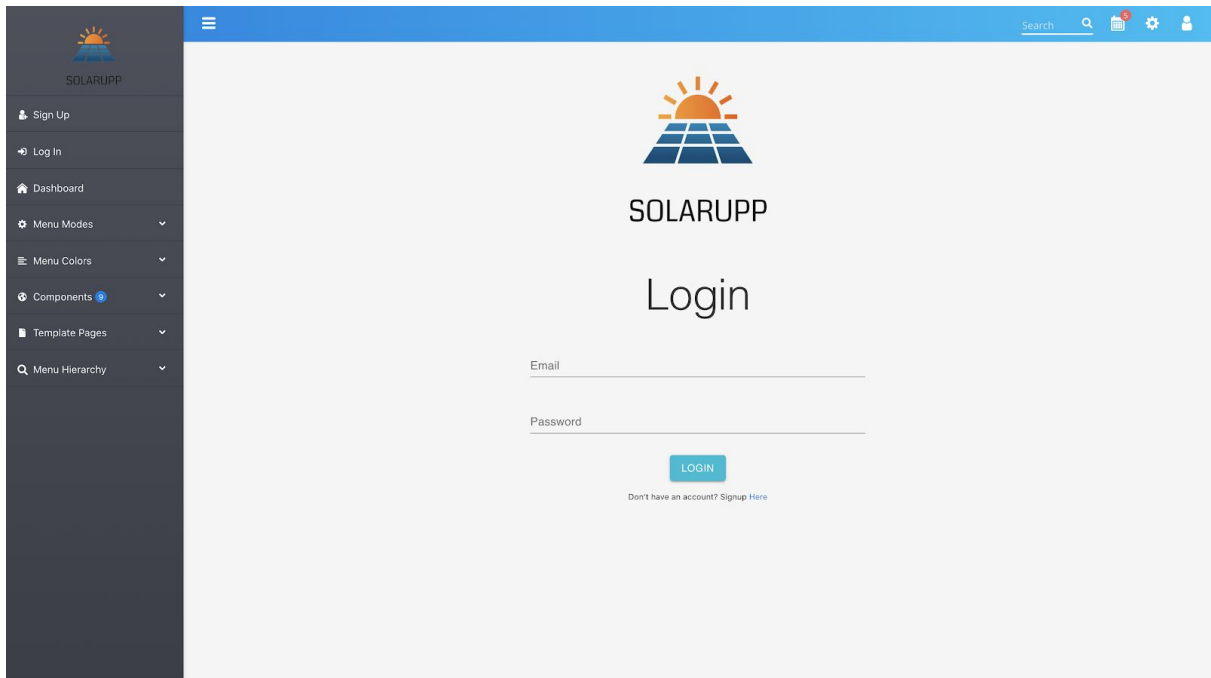


Figure 15: Login screen

This is our login page. After successful login user is directed to Dashboard Page.

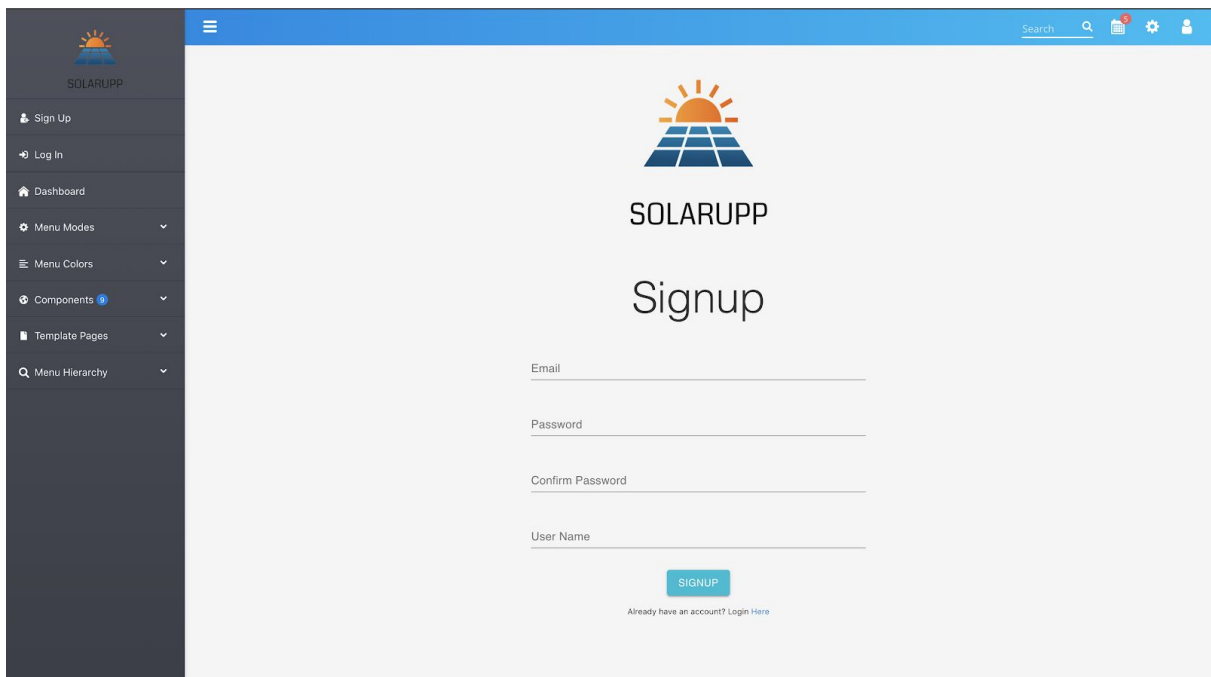


Figure 16: Signup screen

This is our signup page. After successful signup, user is directed to Dashboard page.

☰

Search

Calculate Your Solar Potential

Enter required information to have solar feasibility study.

Your Registered Buildings

Your Buildings

Building Name	Building Type	Address	Roof Area
Home	Apartment		1919.772564248903
Ankuva	Apartment		5361.088373587397
Anane Ev	Apartment		517.1127089366539

⏪

⏩

1

⏪

⏩

Select Building

Choose Your Roof Material

☐ Asphalt Roofs

☐ Wood Shingles

☐ Metal Roofs

☐ Prefabric

☐ Clay

☐ Cement

☐ Slate

Roof Angle

Adjust Your Roof Angle Range: 0,60

⚠ If you know exact angle, please enter in building properties below.

Click to have an idea

Figure 18: Calculate solar potential screen

Users can see their registered buildings. Also they are able to form feasibility studies from here. Users can adjust their roof material and roof angle from here.

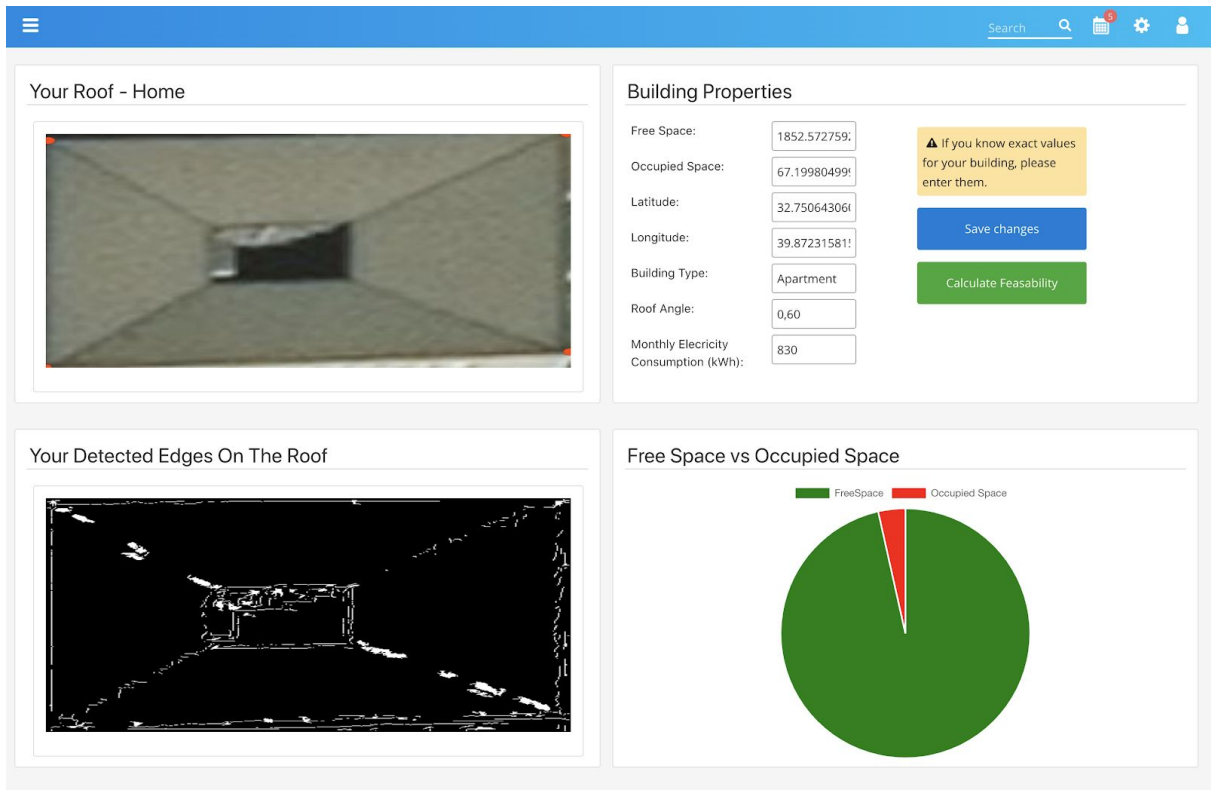


Figure 19: Calculation results screen

Users see their selected roofs in this page, and also see edge-detected version of the rooftop. The pie chart demonstrates free space and occupied space that is calculated from edge-detected version of the rooftop.

Search

5

Compare Plans

Plan 1

Building

Select an address

Plan

Select a plan

Panels

Number of panels

10

Surface

Total surface area of panels

650

System Size

Maximum capacity of panels

4 kW

Annual Production

Estimated average annual production

3500 kWh

Plan 2

Building

Select an address

Plan

Select a plan

Panels

Number of panels

10

Surface

Total surface area of panels

650

System Size

Maximum capacity of panels

4 kW

Annual Production

Estimated average annual production

3500 kWh

Figure 20: Compare plans screen

Users are able to compare their solar plan on this page. After choosing Building and Plan for each Solar Plan, the attributes of plans become visible and can be compared.

Add Existing Solar System to SolarUpp

Your Registered Buildings

Your Buildings			
Building Name	Building Type	Address	Roof Area
example building 3	Suburban Estate	Am Lindenbaum 11, 63927 Bürgstadt, Germany	79.07376314650762
example building 2	Industrial Estate	161 S Wakea Ave, Kahului, HI 96732, USA	3745.5571065947356
uk slumb	Suburban Estate	3 Wilderness Heights, West End, Southampton SO18 3PS, UK	30.67699801222406
adanadaki evim	Suburban Estate	Sehlikler, Kozan Osmaniye Yolu No:960, 80750 Çukurköprü/Kadirli/Osmaniye, Turkey	49.813722063347726
fatcik estate	Apartment	Üniversiteler, İdari Birimler, 06800 Çankaya/Ankara, Turkey	147.36521608177634

⏪ ⏩ 1 ⏪ ⏩

Name of the system:

Number of panels in the system:

Maximum power of the panels (W):

System size (W): 0

Inverter size (W):

Age of the system:

Address:

Country and Region:

Select Country

-

Postal ZIP code:

Angle of the panels:

Save your setup

Figure 21: Add solar system screen

Users are able to add existing solar system to SolarUpp in this page. After that existing solar system data can be tracked from SolarUpp.

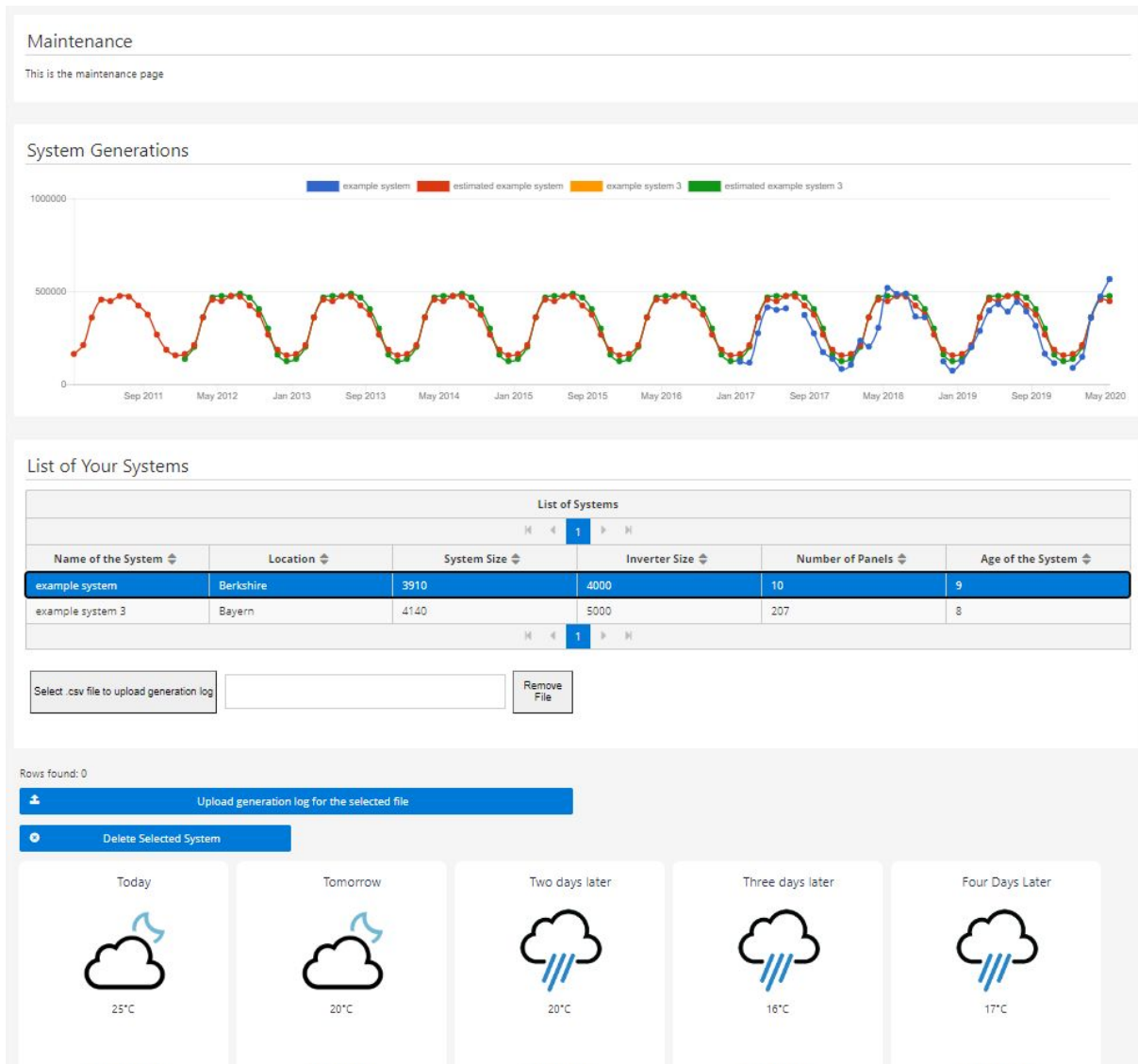


Figure 22: Maintenance screen

Users are able to track their solar panels on this page, and also compare the estimation data with their uploaded data.

Appendix B - Class Diagram

Visual Paradigm Standard (teb@bilkent.edu.tr)

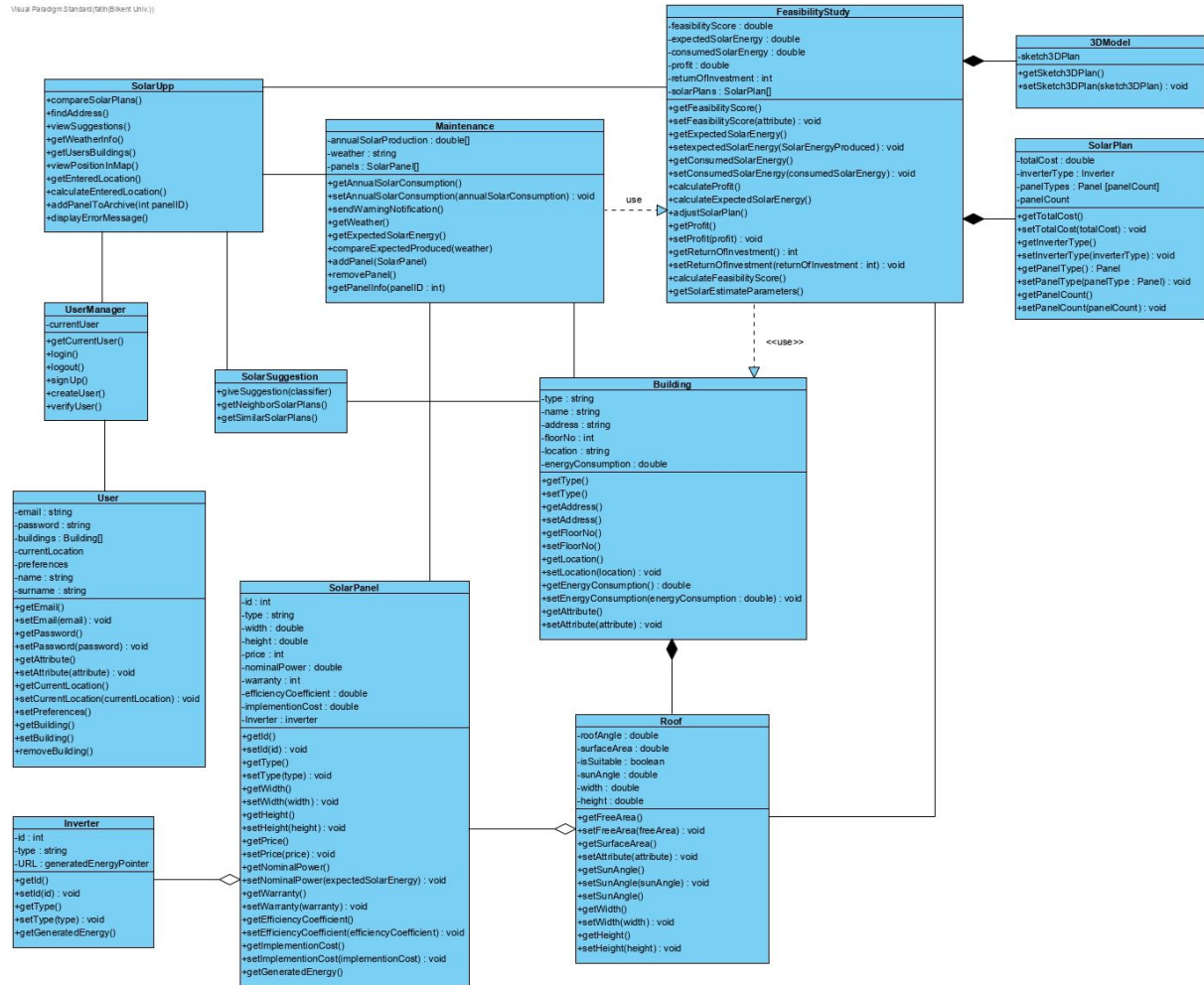


Figure 23: Class diagram

11. References

- [1] "Climate and Energy". http://wwf.panda.org/our_work/climate_and_energy/. [Accessed 26 May 2020].
- [2] Firebase. 2020. *Firestore Realtime Database*. [online] Available at: <https://firebase.google.com/docs/database> [Accessed 26 May 2020].
- [3] Expressjs.com. 2020. *Express - Node.js Web Application Framework*. [online] Available at: <https://expressjs.com/> [Accessed 26 May 2020].
- [4] Cesium. 2020. *Cesium - Changing How The World Views 3D*. [online] Available at: <https://cesium.com/index.html> [Accessed 26 May 2020].
- [5] Resium.darwineducation.com. 2020. *Home*. [online] Available at: <https://resium.darwineducation.com/> [Accessed 26 May 2020].
- [6] GitHub. 2020. *Image-Js/Image-Js*. [online] Available at: <https://github.com/image-js/image-js> [Accessed 26 May 2020].
- [7] Primefaces.org. 2020. *Primereact*. [online] Available at: <https://primefaces.org/primereact/showcase/#/> [Accessed 26 May 2020].
- [8] Google Developers. 2020. *Google Maps Platform | Google Developers*. [online] Available at: <https://developers.google.com/maps/documentation> [Accessed 26 May 2020].
- [9] GitHub. 2020. *BunHouth/Geometry-Library*. [online] Available at: <https://github.com/BunHouth/geometry-library#readme> [Accessed 15 May 2020].
- [10] GitHub. 2020. *Axios/Axios*. [online] Available at: <https://github.com/axios/axios> [Accessed 26 May 2020].
- [11] GitHub. 2020. *Reduxjs/Redux*. [online] Available at: <https://github.com/reduxjs/redux> [Accessed 26 May 2020].
- [12] GitHub. 2020. *Mui-Org/Material-Ui*. [online] Available at: <https://github.com/mui-org/material-ui> [Accessed 26 May 2020].
- [13] OpenWeatherMap.org, "Weather API," *openweather*. [Online]. Available: <https://openweathermap.org/api>. [Accessed 26 May 2020].

- [14] MDN Web Docs. 2020. *Javascript*. [online] Available at: <<https://developer.mozilla.org/en-US/docs/Web/JavaScript>> [Accessed 26 May 2020].
- [15] Google Books. 2020. *Effective Computation In Physics*. [online] Available at: <<https://books.google.de/books?id=DYoNCgAAQBAJ&pg=PA351#v=onepage&q&f=false>> [Accessed 26 May 2020].
- [16] GitHub. 2020. *Build Software Better, Together*. [online] Available at: <<https://github.com/>> [Accessed 26 May 2020].
- [17] Postman.com. 2020. [online] Available at: <<https://www.postman.com/>> [Accessed 26 May 2020].
- [18] Reactjs.org. 2020. *React – A Javascript Library For Building User Interfaces*. [online] Available at: <<https://reactjs.org/>> [Accessed 26 May 2020].
- [19] Node.js. 2020. *Node.Js*. [online] Available at: <<https://nodejs.org/en/>> [Accessed 26 May 2020].
- [20] EU Science Hub - European Commission. 2020. *Photovoltaic Geographical Information System (PVGIS) - EU Science Hub - European Commission*. [online] Available at: <<https://ec.europa.eu/jrc/en/pvgis>> [Accessed 26 May 2020].
- [21] <https://github.com/egeakin/SolarUpp> [Accessed 26 May 2020].
- [22] Shukerullah. 2020. shukerullah/react-geocode. [online] Available at: <https://github.com/shukerullah/react-geocode#readme> [Accessed 26 May 2020]
- [23] Git-scm.com. 2020. *Git*. [online] Available at: <<https://git-scm.com/>> [Accessed 26 May 2020].