Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: SolarUpp

High-Level Design Report

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

Supervisor: Selim Aksoy
Jury Members: Uğur Güdükbay and Jury Abdullah Ercüment Çiçek

High-Level Design Report
Dec 31, 2019

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

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 regarding to 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.

In this report, there is a brief description of our project in terms of high-level design. As an introduction, purpose of the system, design goals, definitions, and overview are explained briefly. Afterward, the current software architecture is being discussed. Then, proposed software architecture is clarified as subsystem decomposition, hardware/software mapping, persistent data management, access control and security, global software control, and boundary conditions. The presentation tier subsystem, application tier subsystem, and data tier subsystem are considered under subsystem services. Additionally, there is a brief description of new knowledge acquired and learning strategies. Lastly, there are glossary and references part at the end of the report.

1.1 Purpose of the system

SolarUpp aims to have a feasibility study on solar energy production and the efficiency of building rooftops in order to increase the usage of green energy. In this web application, solar ratings of rooftops will be calculated by gathering information from images of rooftops. The users will easily find their building's images by entering their addresses. After finding their buildings, users will be able to choose spaces to plant solar panels on their rooftops. In order to calculate solar energy production, users will have the chance to compare different solar panels and inverters to find an optimum solution on their rooftops in terms of cost, size, and efficiency.

Besides a feasibility study on energy production, SolarUpp will also provide a 3D model of a rooftop by visualization to show the user's rooftops' final state after plantation of solar panels and inverters by regarding the real size of solar components and chosen free space on rooftops. To calculate solar energy production in the best way SolarUpp will be able to determine distinct obstacles on rooftops such as pools and air conditioning components to calculate free space on roofs.

After solar panel plantation, generated solar energy amounts will be gathered from registered users' inverters in time intervals like one week. The gathered information will be uploaded to our system to compare the expected and generated value of solar energy. Recent weather conditions in the area of the specified building will be checked to see if there are any additional reasons which cause low energy production than expected rather than unexpected weather conditions. For example, if a rooftop has a great difference between expected and produced energy levels in optimum weather conditions, registered users will be informed with a warning message about the need for maintenance check or an unexpected problem.

SolarUpp is a greenfield project which aims to increase solar energy usage by gathering information from rooftop's images which are taken from above. Using images of rooftops to have a feasibility study is an essential point for large scale using by users

1.2 Design goals

In this section design goals of the SolarUpp will be presented in five subsections. Detailed knowledge about the design goals is explained briefly.

1.2.1 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.

1.2.2 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.

1.2.3 Extendability

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

1.2.4 Scalability

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

1.2.5 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.

1.3 Definitions, acronyms, and abbreviations

API: Application User Interface

JS: JavaScript

ReactJS: React JavaScript Framework

JSON: JavaScript Object Notation

FPS: Frame per second

1.4 Overview

SolarUpp aims to have a feasibility study on solar energy production and the efficiency of building rooftops in order to increase the usage of green energy. In this web application, solar ratings of rooftops will be calculated by gathering information from images of rooftops. The users will easily find their building's images by entering their addresses. After finding their buildings, users will be able to choose spaces to plant solar panels on their rooftops. In order to calculate solar energy production, users will have the chance to compare different solar panels and inverters to find an optimum solution on their rooftops in terms of cost, size and efficiency.

Besides a feasibility study on energy production, SolarUpp will also provide a 3D model of a rooftop by visualization to show the user's rooftops' final state after plantation of solar panels and inverters by regarding the real size of solar components and chosen free space on rooftops. To calculate solar energy production in the best way SolarUpp will be able to determine distinct obstacles on rooftops such as pools and air conditioning components to calculate free space on roofs.

After solar panel plantation, generated solar energy amount will be gathered from registered users's inverters in time intervals like one week. The gathered information will be uploaded to our system to compare the expected and generated value of solar energy. Recent weather conditions in the area of specified building will be checked to see if there are any additional reasons which cause low energy production than expected rather than unexpected weather conditions. For example, if a rooftop has great difference between expected and produced energy levels in optimum weather conditions, registered users will be informed with a warning message about the need for a maintenance check or an unexpected problem.

SolarUpp is a green field project which aims to increase solar energy usage by gathering information from rooftop's images which are taken from above. Using images of rooftops to have a feasibility study is an essential point for large scale using by users

2. Current software architecture

These APIs will be used to calculate the overall feasibility of a building for solar panels. The information accumulated from these APIs will allow us to provide the features of our application.

2.1 Weather API

https://openweathermap.org/api

This weather API provides current and forecast weather data collection. This API can be used for the maintenance service that SolarUpp provides. This is necessary for turnable solar panels, In cloudy weather, it should not work. [2]

2.2 Solar Map API

https://globalsolaratlas.info/map

This API provides how much a specific region gets solar energy. This is a global solar map. This API will be used for calculating expected solar energy. [3]

2.3 Google Maps and Google Earth Engine

https://earthengine.google.com/

https://www.google.com/maps

Google Earth Engine and Google Maps provide visualization of buildings and streets. We will use Google Earth to identify the slope and surface area of rooftops. Also, Google Earth services will be used for a user to find the address of his building and demonstrate it on the map. [4]

2.4 Solar Calculation API

https://solcast.com/

Solcast API provides solar radiation data, historical time series solar irradiance, utility-scale solar forecasts, rooftop solar forecasts, grid aggregations, and utility-scale wind forecasts. This API will be used for our predictions to be more accurate. [5]

2.5 Reality Capture API

https://forge.autodesk.com/api/reality-capture-cover-page/

This API will be used to create a visualization of a user's rooftop after having a feasibility study for selected solar panels as a final state.[7]

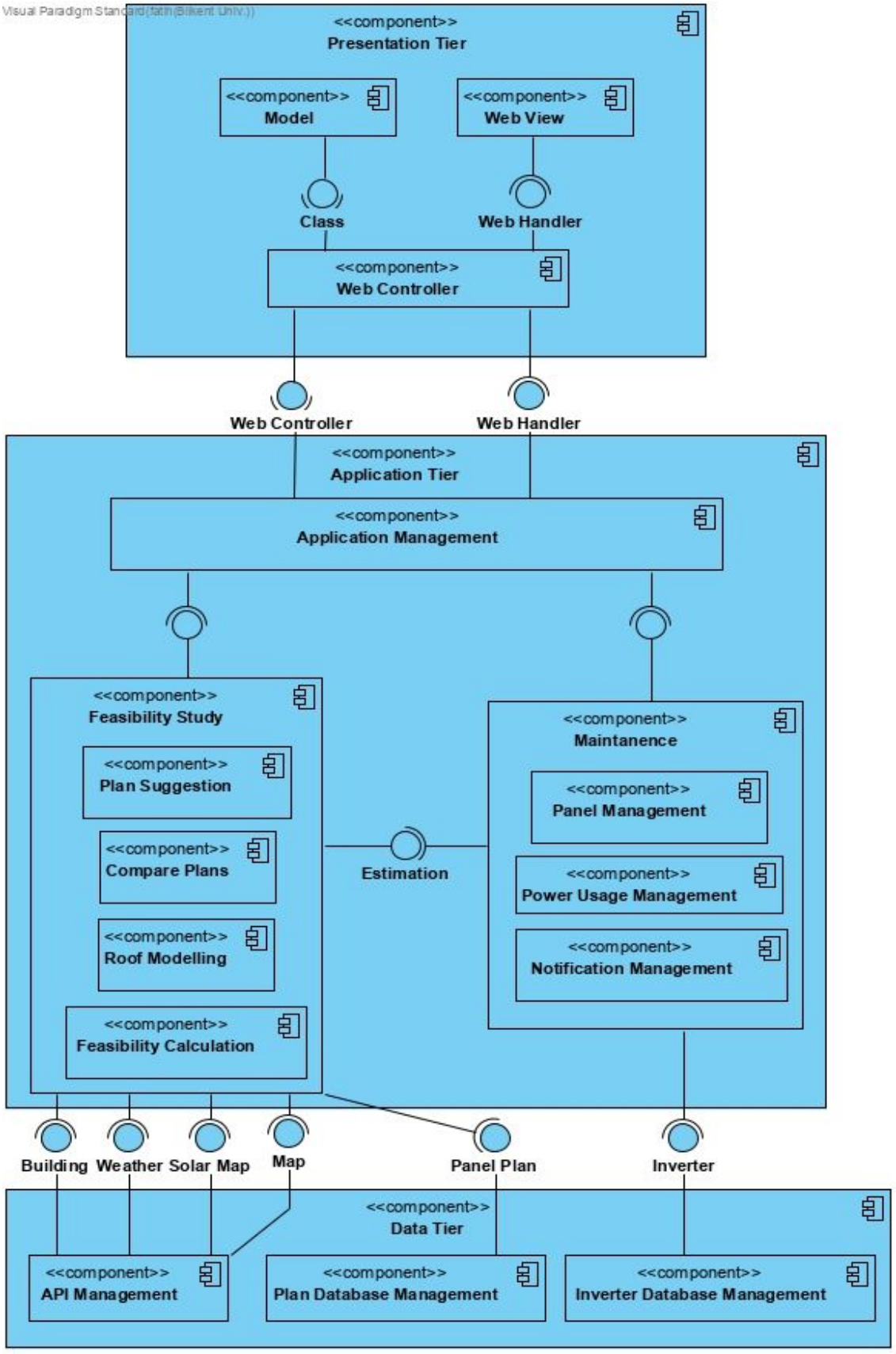
3. Proposed software architecture

3.1 Overview

We defined the design goals of the project, and decompose the system into smaller subsystems that can be realized by individual subgroups. Since our project is a software-intensive project, we also selected strategies for building the system, such as the hardware/software platform on which the system will run, the persistent data management strategy, the global control flow, the access control policy, and the handling of boundary conditions.

We chose 3 Tier Architecture because it enables us to modularize user interface, business logic and data storage independently. Therefore it provides flexibility for the implementation stage.

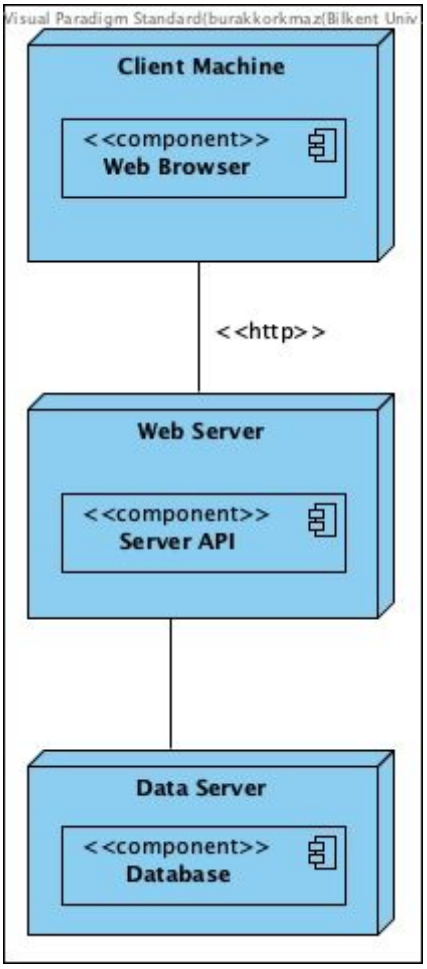
3.2 Subsystem decomposition



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 such as openWeatherMap, Solcast, etc. 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.

3.3 Hardware/software mapping



SolurUpp will be designed by following principles of 3-Tier Architecture.Client and server software will be divided into tiers in our application. The system will only have one client which is a web browser where users will run it on their computers. Clients will connect to the server by using the HTTP protocol. In the presentation tier, client software will take place and it will handle both controller and view subsystem of our application. In middle tier, server software takes place and it will contain business layer functionalities of our application such as feasibility study, modelling, etc. Server software runs on the application server in order to handle client requests. Data tier will have a NoSqlDatabase component. It will store user specific data and data for business layer functionalities such as weather information or solar maps. Components in data tier and presentation tier can

optionally run on the same device or different devices by considering performance issues. MongoDB will be used as NoSqlDatabase to perform text based searches rather than SQL queries. In the application tier React framework will be used to develop a web application on client side.

3.4 Persistent data management

SolarUpp is going to store the data of the solar panels, feasibility studies, personal data, and building information. Apart from that, cookies will not be used for persistent data. To satisfy persistent data storage, we will use the Firebase database, a NoSQL cloud database system in which data is stored across all clients in realtime, powered by Google. Data will be stored as JSON which will allow storing images, building models, and necessary files. All the stored data will be synchronized in real-time.

3.5 Access control and security

Since we will use the Firebase Database in SolarUpp, we can save passwords cryptic. Firebase Open Source has Firebase Authentication Password Hashing, which uses a script to hash passwords. Firebase generates a unique hash parameter to encode passwords. So, It will rehash the password in the first successful login. Users will access their accounts by their email addresses and passwords. The security of the accounts will be saved by encoding their passwords in our database system.

3.6 Global software control

Global software control implemented in a centralized way in SolarUpp. Software control is dependent on the actions of users and their requests. SolarUpp provides and uses different information in order to have a feasibility study for a rooftop. Solar maps of different regions, up to date and old weather information for these regions can be examples of information will be used to have a solar feasibility study. This information will be gathered by using different APIs described in the

second section. Reliability is one of the most important design goals of SolarUPP in order to have an accurate solar feasibility study for users. To achieve this goal, gathered data should be reliable and up to date so the server will have different API calls to gather data and stabilize the reliability of the information that will be used. Implemented global software control of services will be asynchronous and event driven.

If a user requests to have a feasibility study for the first time, information specific to the position of the targeted roof will be considered. Data like solar radiation values, solar maps, weather information, etc in order to have a feasibility study will be fetched from our NoSQL database and required calculations will take place on the server side of SolarUpp. To success in reliability, updates for required information will take place in predetermined time periods. Since SolarUpp will use Google Maps API for roof imagery for satellite images there will be no need for updates related to the imagery of rooftops.

3.7 Boundary conditions

There are three boundary conditions in our proposed software architecture: initialization, termination, and failure.

3.7.1 Initialization

Users will need a web browser to access SolarUpp. An internet connection is necessary to operate SolarUpp. To use our application, they have to enter their email address and password to access their accounts. In case of entering a wrong email address or password, there will be an error message. If a potential user does not have an account, he needs to enter name, surname, a valid email address, and password to sign up SolarUpp. If he enters a nonvalid email address an error message will appear.

3.7.2 Termination

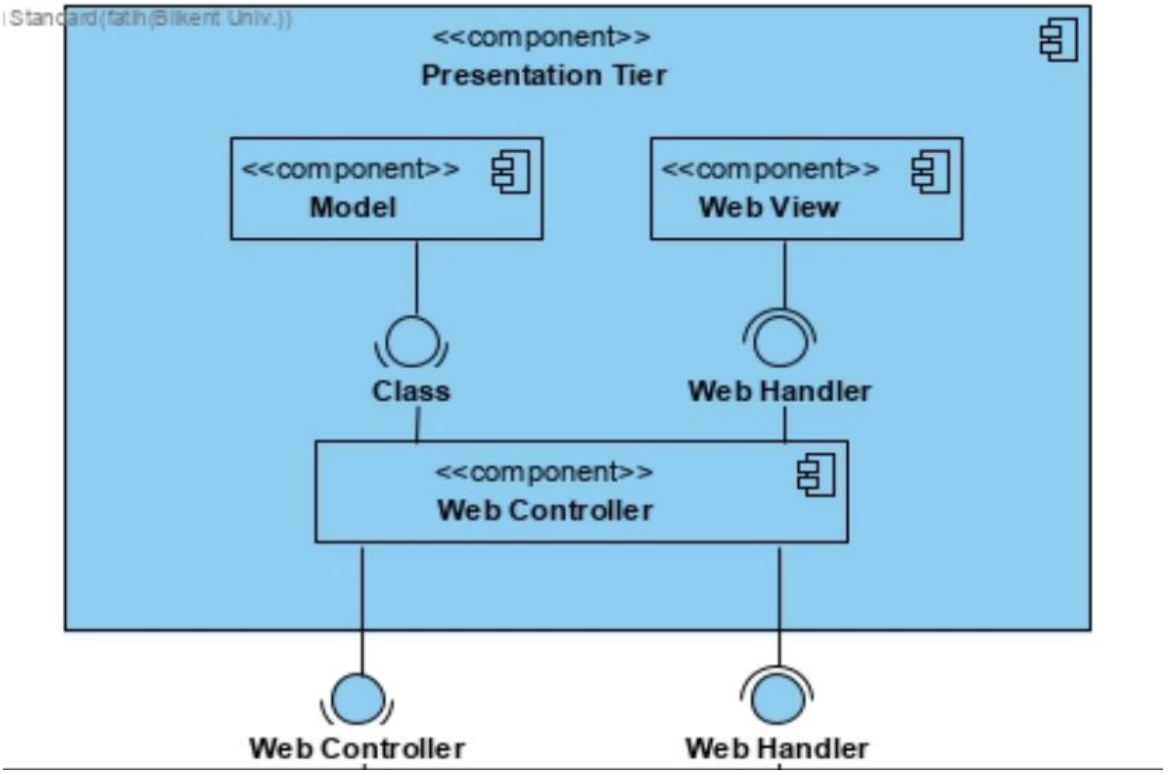
Users will terminate their section by logging out from the application on their web browser.

3.7.3 Failure

In case of an overload to our servers, users may not access our application. Other than that, due to the traffic between the third party API's and SolarUpp, there might be some connection problems. Moreover, the internet speed of users may cause latency on Firebase connection with the client side.

4. Subsystem services

4.1 Presentation Tier Subsystem



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

4.1.1 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.

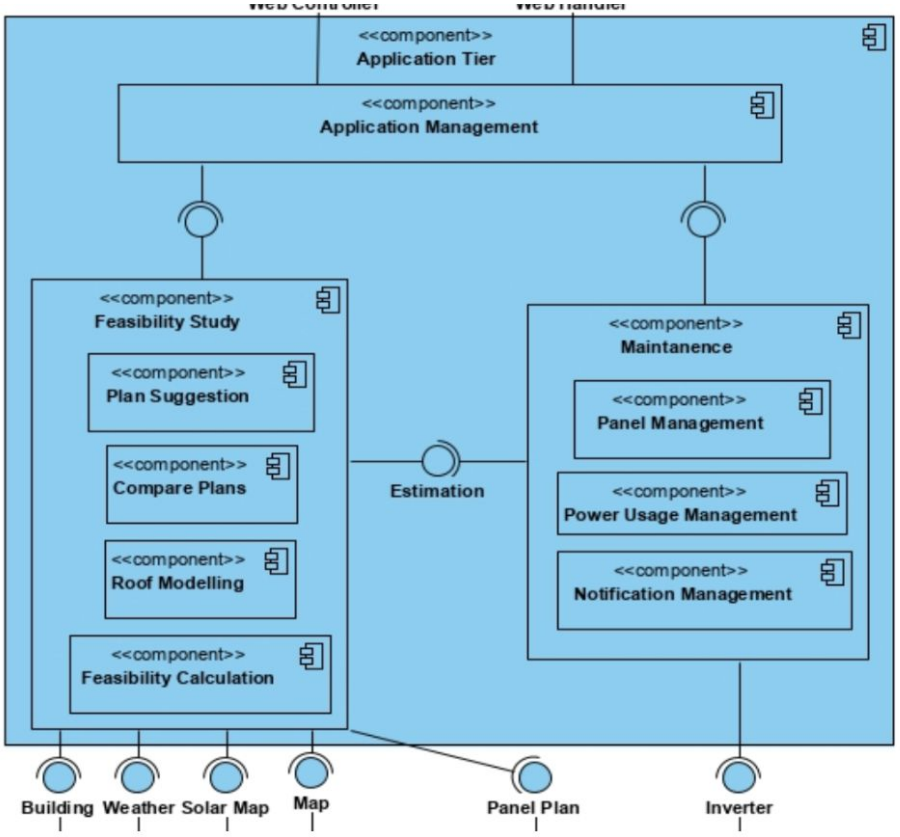
4.1.2 Web Controller

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.

4.1.3 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.

4.2 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.

4.2.1 Application Management

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

4.2.2 Feasibility Study

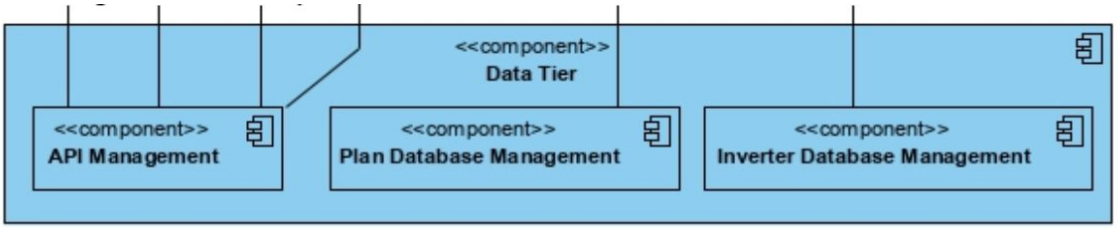
Feasibility Study Component contains 4 subsystems in itself. These subsystems are the following: Plan Suggestion, Compare Plans, Roof Modelling, and Feasibility Calculation. Plan Suggestion component provides the Solar Plan Suggesting service that SolarUpp provides. This component

contains machine learning models in itself and according to the machine learning model and datasets about solar panels, Plan Suggestion component suggests the most suitable Solar Plan Model to the client's roof. 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. Feasibility Calculation component provides all calculations such as feasibility score, estimation of solar energy produced, estimation of net profit. To implement the mentioned components, Python will be used.

4.2.3 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 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, Python will be used.

4.3 Data Tier Subsystem



The back-end services of SolarUpp are provided in 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 will interactively manage the database side of our application.

4.3.1 API Management

This component will handle the API requests for our application. For more information on APIs, please refer to section 2 (Current software architecture).

4.3.2 Plan Database Management

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

4.3.3 Inverter Database Management

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

5. 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 require 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.

6. 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

7. References

[1] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.

[2] OpenWeatherMap.org, "Weather API," *openweather*. [Online]. Available: https://openweathermap.org/api. [Accessed: 10-Nov-2019].

[3] Solargis, *Global Solar Atlas*. [Online]. Available: https://globalsolaratlas.info/map. [Accessed: 10-Nov-2019].

[4] "Solar Forecasting & Solar Irradiance Data," *Solcast*. [Online]. Available: https://solcast.com/. [Accessed: 10-Nov-2019].

[5] *Google Maps Platform | Google Developers*. [Online]. Available: https://developers.google.com/maps/documentation. [Accessed: 10-Nov-2019].

[6] *Google Earth Engine*. [Online]. Available: https://earthengine.google.com/. [Accessed: 10-Nov-2019].

[7] C. Johnson, "PSMA Buildings," *ProgrammableWeb*, 18-Dec-2018. [Online]. Available: https://www.programmableweb.com/api/psma-buildings. [Accessed: 10-Nov-2019].

[8] "Reality Capture API," *Reality Capture Cover Page | Autodesk Forge*. [Online]. Available: https://forge.autodesk.com/api/reality-capture-cover-page/. [Accessed: 10-Nov-2019].