Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: SolarUpp

Low-Level Design Report

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

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

Low-Level Design Report
Feb 17, 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

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.

In this report, there is a brief description of our project in terms of low-level design. As an introduction, the design trade-offs are discussed followed by an explanation of guidelines and standards used. Afterward, the packages of the system and the class interfaces are explained. Lastly, there are glossary and references part at the end of the report.

1.1 Object Design Trade-offs

The trade-offs between design goals in our design are discussed here.

1.1.1 Usability vs. Functionality

SolarUpp supports various services. These services are feasibility study, maintenance, 3D solar plan modelling, suggestion system, etc. Since we have that many services, for now we choose functionality over usability. When we looked at other software platforms that are similar to our services, we did not find any platform which combines maintenance of solar plans and feasibility study. The innovative aspect of our project is combining these 2 main services. Therefore SolarUpp offers a lot of functionality and while providing these functionalities, usability might be decreased.

1.1.2 Cost vs. Supportability

The initial SolarUpp demo will be only supported over web browsers. The reason behind this design decision is to cut down development time. Perhaps the further versions will be supported on the mobile devices too. On the other hand, SolarUpp needs stable server connection to function, and the requests to the server have to be handled in a respective manner. Hence, it is crucial to establish reliable servers with stable connections which will increase the cost of the application.

1.1.3 Space vs. Time

For many web applications, from users perspective using the application in a fluent and fast way is one of the most important points. SolarUpp stores many data such as detailed information about solar panels, inverters, past weather information

and solar radiation information of different locations around the world. SolarUpp stores many of this data on the server side in order to have less memory usage in clients devices. In order to operate the data exchanges between server side and client side more time will be spent on to ensure a fluent user experience in SolarUpp.

1.1.4 Scalability vs. Performance

SolarUpp aims to be used by a great number of users around the world to have their solar energy feasibility study. Performance is one of the most important design goals for any web application. However, dealing with a great number of requests simultaneously for a high number of users will cause high workload on the server side. Both serving a great number of clients and providing a high performance for users will be a challenging process.

1.2 Interface documentation guidelines

- Class names and methods are written in ‘Camel Case’ format.
- Functions will be written with parentheses if it has parameters, there will be no parentheses if there are no parameters. Such as ‘method1(parameter1, parameter20) and ‘method2’.
- Both class attributes and methods will be described as public or private.

Class Representation format in this report:

Class ClassName	
Brief class description.	
Attributes	
private attributeType attribute1	Description of attribute1
public attributeType attribute2	Description of attribute2
Methods	
public returnType method1WithNoParameters	Description of method1WithNoParamaters
private returnType method2WithParameters(parameterType paramater)	Description of method2WithParameters

1.3 Engineering standards (e.g., UML and IEEE)

1.3.1 UML Notation

To show package diagrams and relations between components and classes UML Notation is used in this report.

1.3.2 IEEE Standards

For citation and references IEEE standards used in this report.

1.4 Definitions, acronyms, and abbreviations

API: Application User Interface

JS: JavaScript

ReactJS: React JavaScript Framework

JSON: JavaScript Object Notation

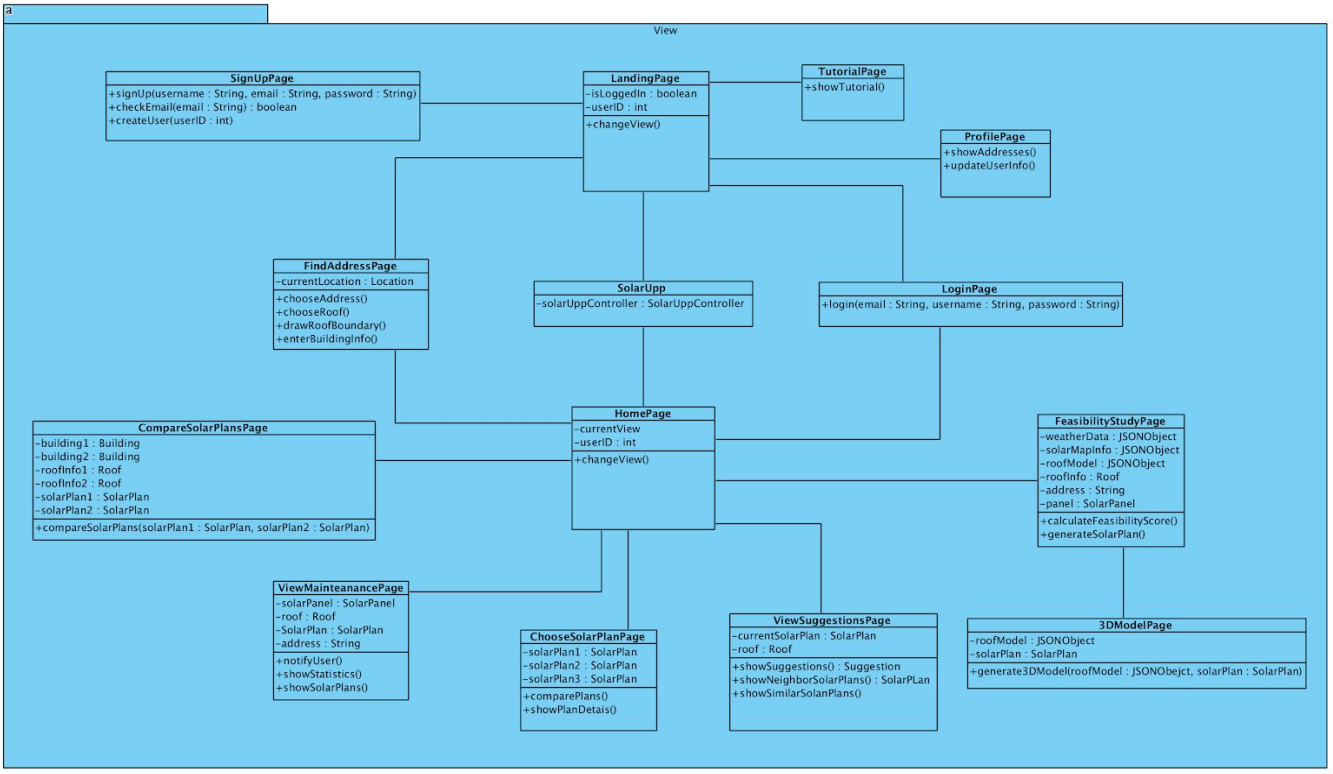
FPS: Frame per second

2. Packages

2.1 Client Package

2.1.1 View Package

View Package provides the user interface interactions. It serves as the front-end of the SolarUpp application.



- **SignupPage:** This page is the page that unregistered users register the system.

- **LandingPage:** LandingPage is the initial page that all users see when he opens SolarUpp.

- **TutorialPage:** This page contains a small video tutorial for using the software.

- **ProfilePage:** ProfilePage is the page that demonstrates the user's information.

- **FindAddressPage:** This page contains a map and users are able to find his address within this map.

- **SolarUpp:** This is the main class that manages the transfer between pages.

- **LoginPage:** This page is the page that verified users are able to login to the system.

- **CompareSolarPlanPage:** This page is the page that users are able to compare his buildings' solar plans and decides which plan is the efficient one.

- **HomePage:** This page is the base in which users are able to go all other pages such as chooseSolarPlanPage, FindAddressPage, etc.

- **FeasibilityStudyPage:** This page mainly shows the calculations of the feasibility study that is done on the user's building's rooftop.

- **ViewMaintenancePage:** This page shows the maintenance information of the solar plan of the user and users are able to see whether there is a defect on the functionality of his solar plan or not.

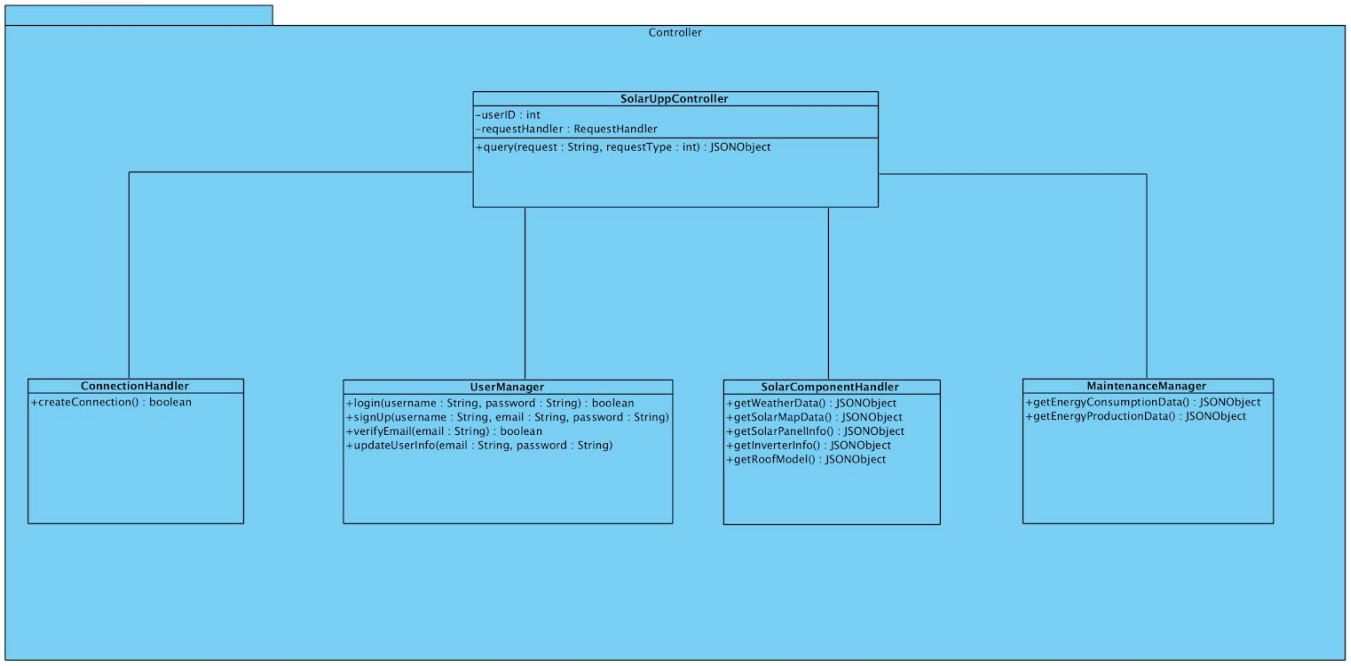
- **ChooseSolarPlanPage:** This page will provide all available plans for the selected address to the users.

- **VIewSuggestionsPage:** This page demonstrates the available solar plans to building within the SolarUpp which is similar to the user's building.

- **3DModelPage:** This page demonstrates the model of solar panel plan on users' buildings in 3D.

2.1.2 Controller Package

Controller Package provides some low-level system functionalities such as user verification, establishing the connection to the server, data transfer between a logic package in order to meet the user's requests. This package basically controls screen swaps and data transfer between classes in View Package.



- **SolarUppController:** SolarUppController provides communication between View Package and Logic Package. It manages the user requests and transmits these requests to the logic package.

- **ConnectionHandler:** ConnectionManager provides connection to the SolarUpp server.

- **UserManager:** UserManager handles the user information updates that are coming from the view package and reflects these modifications to the logic package.

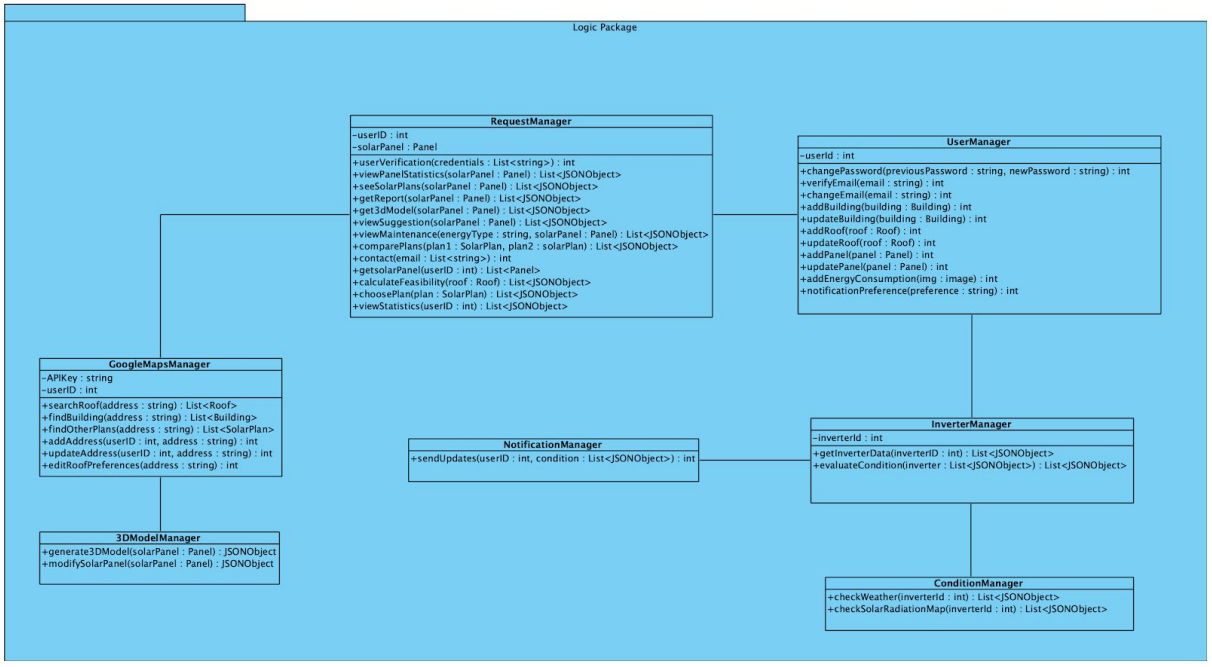
- **SolarComponentHandler:** When a user creates a feasibility study on his solar panel, he automatically invokes the needed information that is used on calculating predicted energy production. This handler transmits these requests to logic data and gets calculation data such as weatherData, solarMapData, InverterInfo, etc.

- **MaintenanceManager:** MaintenanceManager receives the user requests for his solar panel maintenance and transmits this request and gets the energy consumption and energy production data from the logic package.

2.2 Server Package

2.2.1 Logic Package

Logic package provides the handling of requests from the client and within the application packages. Briefly, it includes connection logic of server side and client side.



- **Request Manager**: This class handles the requests of the server and client sides from the View Package. It gets requests from the View Package and it sends them back to the View Package.

- **User Manager**: This class includes user preferences. Users can verify, add or edit their preferences such as email address, password, notification settings. Besides personal preferences, users can change their building, roofs, and solar panels. Moreover, energy consumption data can be added to predict future incomes.

- **Inverter Manager**: This class gets the data from inverters and evaluates the conditions of the inverter to report the process to the user.

- **Google Maps Manager**: This class is responsible for the connection between the Google Maps API and the server side. 2D models of the cities will be provided by that class for the users to select their roofs to install solar panels. Moreover users can discover other solar panels located nearby.

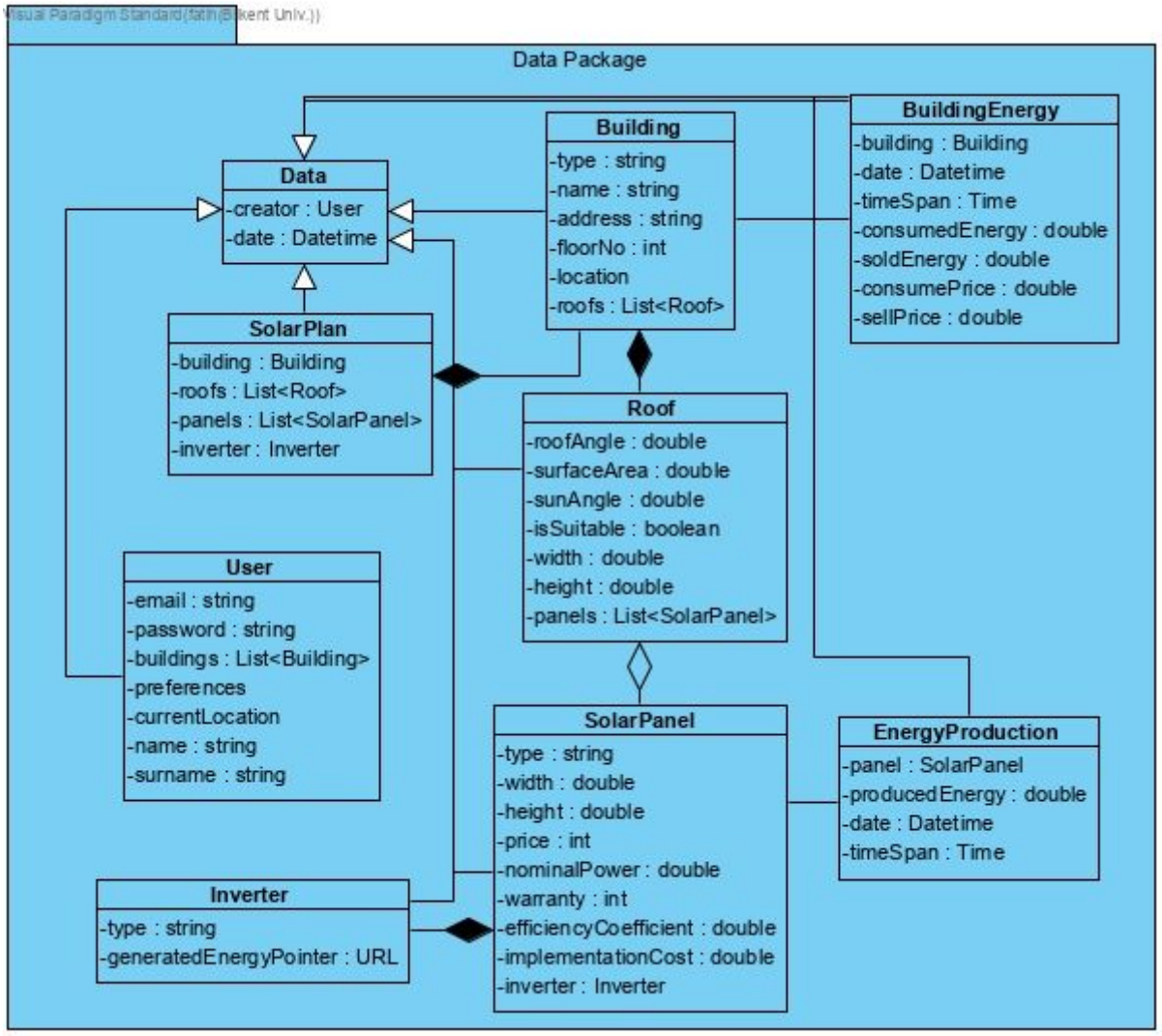
- **3D Model Manager**: This class converts the 2D model of the roof and solar panels into 3D models to provide a better virtual view for the users.

- **Condition Manager**: This class provides weather and solar radiation information from API's to the selected roofs. The inverter manager uses that information to predict feasibility.

- **Notification Manager**: This class handles notification services for users in an emergency situation. Moreover it provides instantaneous information about panels to the View Package.

2.2.2 Data Package

The data package consists of the database elements. Only the logic package can interact with the data package.



- **Data**: The Facade class of the package. Every class in the package is a child class of Data class. It stores the creator user and the creation date.

- **User**: This class stores information about the users.

- **SolarPlan**: This class stores the user made solar plans. Used for the feasibility study of a building, it has no use for maintenance. It enables users to save their progress while creating a plan.

- **Building**: This class is responsible for holding information about buildings. Building class has roof objects because a single building may have several roofs.

- **BuildingEnergy**: This class is used to store the past statistics about a building. It stores both how much electricity is bought from the grid and how much is sold.

- **Roof**: This class contains all of the features of the roofs such as roof angle, sun angle, width, height, and surface area. These features will be used for parameters of calculating the feasibility score of the rooftops and estimated solar energy production.

- **SolarPanel**: This class is used for identifying every solar panel registered in the application. It has some variables such as id, type, width, height, price, nominal power, warranty, efficiency coefficient, implementation, cost, and inverter type which will be used for the maintenance service.

- **EnergyProduction**: This class is used to store the past statistics of generated electricity for panels.

- **Inverter**: This class holds general information and properties of an inverter.

3. Class Interfaces

In this section, the classes in the packages will be identified briefly with purposes, attributes and methods. Moreover, the relation between packages and methods will be mentioned.

3.1 Client Package

3.1.1 View

Landing Page	
This page is the page that unregistered users register the system.	
Attributes	
private int userID	Unique User ID that is generated for each user.
private boolean isLoggedIn	This boolean value checks whether the current user is successfully logged in or not
Methods	
public void changeView	This method is for changing pages.

SignUp Page	
This page is the page that unregistered users register the system.	
Methods	
public void signUp(String Username, String email, String password)	This method is for signing up for the new user and this method requests new user in the database.

public boolean checkEmail(String email)	This method is for user verification via email.
public void createUser(int userID)	This method creates a user with unique user ID.

Tutorial Page	
This page is to play the tutorial video to teach how to use the app.	
Methods	
public void showTutorial	This method opens and plays the tutorial of the usage of SolarUpp

Profile Page	
ProfilePage is the page that demonstrates the user's information.	
Methods	
public void showAddress	This method is for showing user's all addresses that are entered to the SolarUpp system.
public void updateUserInfo	This method updates the user credentials.

FindAddress Page	
This page contains a map and users are able to find his address within this map.	
Attributes	
private Location currentLocation	This attribute is a default currentLocation, and if it not changed solar plan will be done for this current location.
Methods	
public void chooseAddress	This method is for user picking and marking his own address on the map.
public void chooseRoof	This method is for user picking and marking his own roof on the map.
public void drawRoofBoundary	This method is for user be able to draw the boundaries of solar panel on his rooftop.
public void enterBuildingInfo	This method is for user entering information about building manually if needed.

SolarUpp	
This is the main class that manages the transfer between pages.	
Attributes	
private SolarUppController solarUppController	This attribute is providing communication between low-level packages with View Package.

Login Page	
This page is the page that verified users are able to login to the system.	
Methods	
public boolean login(String email, String username, String password)	This method is for pre-registered users to log in to the system.

CompareSolarPlan Page	
This page is the page that verified users are able to login to the system.	
Attributes	
private Building building1	This attribute is the first building that is going to be compared.
private Building building2	This attribute is the second building that is going to be compared.
private Roof roofInfo1	This attribute is the first roof that is going to be compared.
private Roof roofInfo2	This attribute is the second roof that is going to be compared.
private SolarPlan solarPlan1	This attribute is the first solar plan that is going to be compared.
private SolarPlan solarPlan2	This attribute is the second solar plan that is going to be compared.
Methods	
public void compareSolarPlan	This method is for comparing solar panel panels on users' multiple buildings.

ViewMaintenance Page	
This page shows the maintenance information of the solar plan of the user and users are able to see whether there is a defect on the functionality of his solar plan or not.	
Attributes	
private SolarPanel solarPanel	SolarPanel attribute is for representing the user's solar panel that is used for maintenance service.
private Roof roof	Roof attribute is for representing user's roof that is used for maintenance service.
private SolarPlan solarPlan	SolarPlan attribute is for representing the user's SolarPlan that is used for maintenance service.
private String address	Address attribute is for representing the user's address that is used for maintenance service.
Methods	
public void notifyUser	This method notifies the user via email if there is a defect on the functionality of the solar panel.s
public void showStatistics	This method shows statistics of Solar Plan such as energy produced with respect to months.
public void showSolarPlans	This method shows all of the solar plans of the user.

Home Page	
This page is the base in which users are able to go all other pages such as chooseSolarPlanPage, FindAddressPage, etc.	
Attributes	
private currentView	This attribute demonstrates the current

	page which is rendered to the user.
private userID	This attribute transferred between pages to see which user is currently active.
Methods	
public void changeView()	This method will be used for changing pages and rendering new views for presentation.

ViewSuggestions Page	
This page demonstrates the available solar plans to building within the SolarUpp which is similar to the user's building.	
Attributes	
private SolarPlan currentSolarPlan	This attribute holds the user's solar plan if the user formed a solar plan before and comparison with other suggestions will be made according to this attribute.
private Roof roof	The roof attribute is for finding some efficient solar plan suggestions for the user's rooftop.
Methods	
public Suggestion showSuggestions	This method show suggestions made by SolarUpp.
public SolarPlan showNeighborSolarPlans	This method shows solar plans which are on the neighborhood of the user's building.
public SolarPlan [] showSimilarSolarPlans	This method shows solar plans which are made on the similar roof of the user.

FeasibilityStudy Page	
This page mainly shows the calculations of the feasibility study that is done on the user's building's rooftop.	
Attributes	
private JSONObject weatherData	weatherData attribute is coming from API which is for calculating more precise solar panel feasibility and solar energy production estimate.
private JSONObject solarMapInfo	SolarMapInfo attribute is coming from an open-source API and used for precise feasibility calculations.
private Roof roofInfo	roofInfo attribute holds information about the roof of the user.
private String address	This attribute holds the address of the user
private SolarPanel solarPanel	SolarPanel attribute holds the solar panel that is on the solar plan of the user
Methods	
public double calculateFeasibilityScore	This method calculates the feasibility score of the user's solar plan. This method needs a lot of parameters such as coordinates, roof information, solar map of the coordinates, etc. By using multiple parameters that mentioned above feasibility score is calculated.
public SolarPlan [] generateSolarPlan	This method generates multiple efficient solar plans after finishing the feasibility study on the user's roof.

View3DModel Page	
This page demonstrates the model of solar panel plan on users' buildings in 3D.	
Attributes	
JSONObject roofModel	This indicates the roof of the user, and it is also used in 3D model of the solar plan.
SolarPlan SolarPlan	SolarPlan indicates the user's plan that is going to be modeled.
Methods	
public void generate3DModel(JSONObject roofModel, SolarPlan solarPlan)	This method generates 3D model of user's solar plan.

3.2.1 Controller

SolarUppController	
SolarUppController provides communication between View Package and Logic Package. It manages the user requests and transmits these requests to the logic package.	
Attributes	
private int userID	Operations in business logic of application needs data exchange between server and client side. This data can be transferred if the user was successfully logged into the system and userID represents current online user.
private Request Handler requestHandler	Data exchange between client and server side in SolarUpp done with queries and requests. This attribute resembles the server side of connection created to operate data transfer with query string. With this attribute we can

	get already existing data and create new data.
Methods	
public JSONObject query(String request, int requestType)	This method is used for data operation between client and server side by having a string request and a request type. Request type of data operations can be ‘GET’, ‘UPDATE’, ‘DELETE’ which are SQL based data operations. Specified method returns the requested data and its result as a JSONObject.

ConnectionHandler	
ConnectionManager provides connection to the SolarUpp server.	
Methods	
public boolean createConnection	Information needed to operate a connection between client and server side such as port number will be coded in a xml file in application. This method establishes a connection between server and client side for web application.

UserManager	
UserManager handles the user information updates that are coming from the view package and reflects these modifications to the logic package.	
Methods	
public boolean login(String username, String password)	Check user credentials and returns a boolean value due to matching

	username and a password.
public boolean signUp(String username, String password)	With given username, email and password user will register to system as a new user if current username is not taken.
public boolean verifyEmail(String email)	Email taken by parameter will be checked if it is valid or not and result will be returned as a boolean value.
public void updateUser(String email, String password)	This method will be used for updating user specific information.

SolarComponentHandler	
When a user creates a feasibility study on his solar panel, he automatically invokes the needed information that is used on calculating predicted energy production. This handler transmits these requests to logic data and gets calculation data such as weatherData, solarMapData, InverterInfo, etc.	
Methods	
public JSONObject getWeatherData	This method returns the weather information of current location which feasibility study is made or related weather information for logged in users. Weather information will be extracted from other web sites and returned as JSONObject.
public JSONObject getSolarMapData	This method gets information about solar radiation values of related location/address in order to operate a feasibility study and check for potential solar energy production. Solar radiation information will be gathered from different sources dynamically and it will be returned to the user as a JSONObject.
public JSONObject getSolarPanelInfo	In order to present several solar plans to users, different solar panels should

	be checked to choose the optimal one. This method gets all details of specified solar panel information which will be required for feasibility study and important for solar panel plantation.
public JSONObject getInverterInfo	This method returns all information of an inverter such as efficiency, potential loss, etc. To choose between different solar plans and products users need to get detailed information about inverters.
public JSONObject getRoofModel	This method returns the current address' image as a model to create a 3D model of the user's roof to demonstrate the upcoming scene of roof.

MaintenanceManager	
MaintenanceManager receives the user requests for his solar panel maintenance and transmits this request and gets the energy consumption and energy production data from the logic package.	
Methods	
public JSONObject getEnergyConsumptionData	This method will be used to get user's electricity consumption in order to calculate upcoming savings and time for return of investment.
public JSONObject getEnergyProductionData	This method returns information about the amount of energy produced by solar panels. Inverters have an interface in order to share data solar energy production. This method will get data from inverter by using its interface dynamically with xml files or file transfer protocol and will be returned as JSONObject.

3.2 Server Package

3.2.1 Logic Package

Request Manager	
This class handles the requests of the server and client sides from the View Package. It gets requests from the View Package and it sends them back to the View Package.	
Attributes	
private int userID	Unique User ID that is generated for each user.
private Panel solarPanel	Unique solar panel with its inverter and roof information.
Methods	
public int userVerification(List<string> credentials)	Checks whether a user is verified by looking at credentials.
public List<JSONObject> viewPanelStatistics(Panel solarPanel)	Stands for instant statistics about a solar panel.
public List<JSONObject> seeSolarPlans(Panel solarPanel)	Stands for available solar plans for selected roof.
public List<JSONObject> getReports(Panel solarPanel)	Gives consumption and productivity reports of a solar panel.
public List<JSONObject> get3dModel(Panel solarPanel)	Gets 3D model of the selected solar panel from 3D Model Manager class.
public List<JSONObject> viewSuggestions(Panel solarPanel)	Views suggestion about selected solar plans by using machine learning techniques.
public List<JSONObject> viewMaintenance(string energyType,Panel solarPanel)	Stand for viewing the maintenance of a solar panel. Energy consumption level can be defined in this method.
public List<JSONObject> comparePlans(solarPlan	Compares possible plans for selected roofs. Two solar plans can be viewed

plan1,solarPlan plan2)	with a comparison report.
public int contact(List<string> email)	This method handles the contact section. Emails from anyone will be delivered to the server.
public List<Panel> getSolarPanel(int userID)	Gives all solar panels of a user.
public List<JSONObject> calculateFeasibility(Roof roof)	Calculates feasibility of a roof by considering all conditions and possible solar panel models.
public List<JSONObject> choosePlan(SolarPlan plan)	Chooses a solar plan for a user.
public List<JSONObject> viewStatistics(int userId)	Views the statistics of a user for the SolarUpp usage.

User Manager	
This class includes user preferences. Users can verify, add or edit their preferences such as email address, password, notification settings. Besides personal preferences, users can change their building, roofs, and solar panels. Moreover, energy consumption data can be added to predict future incomes.	
Attributes	
private int userID	Unique User ID that is generated for each user.
Methods	
public int changePassword(string previousPassword, string newPassword)	Changes the password of a user.
public int verifyEmail(string email)	Verifies the email address of a user.
public int changeEmail(string email)	Changes the email of a user.
public int addBuilding(Building building)	Adds a building address for a user
public int updateBuilding(Building building)	Updates the building address of a user.

public int addRoof(Roof roof)	Adds a roof section which is available for a user.
public int updateRoof(Roof roof)	Updates the roof section which is available for a user.
public int addPanel(Panel panel)	Adds a solar panel on a roof for a user.
public int updatePanel(Panel panel)	Updates a solar panel on a roof for a user.
public int addEnergyConsumption(image roof)	Adds energy consumption of the building with images.
public int notificationPreference(string preference)	Adjusts notification preferences of a user.

Inverter Manager	
This class gets the data from inverters and evaluates the conditions of the inverter to report the process to the user.	
Attributes	
private int inverterID	Unique inverter ID that is generated for each solar panel.
Methods	
public List<JSONObject> getInverterData(int inverterID)	Gets the solar panel data from the inverter.
public List<JSONObject> evaluateCondition(<JSONObject> inverter)	Evaluates the condition of a solar panel in order to send a notification to a user.

Google Maps Manager
This class is responsible for the connection between the Google Maps API and the server side. 2D models of the cities will be provided by that class for the users to

select their roofs to install solar panels. Moreover users can discover other solar panels located nearby.	
Attributes	
private string APIKey	API Key is required to get queries from Google Maps and Google Earth API's on Google Cloud.
private int userID	Unique User ID that is generated for each user.
Methods	
public List<Roof> searchRoof(string address)	Searches available roofs on an address.
public List<Building> findBuilding(string address)	Finds building on a selected address.
public List<SolarPlan> findOtherPlans(string address)	Locates other solar plans at nearby addresses.
public int addAddress(int userID, string address)	Adds address for a user.
public int updateAddress(int userID, string address)	Updates address of a user.
public int editRoofPreferences(string address)	Edits roof information such as angle and direction.

3D Model Manager	
This class converts the 2D model of the roof and solar panels into 3D models to provide a better virtual view for the users.	
Methods	
public JSONObject generate3DModel(Panel solarPanel)	Generates 3D model of a roof and solar panel.
public JSONObject modifySolarPanel(Panel solarPanel)	Modifies the position of a solar panel on a roof.

Condition Manager	
This class provides weather and solar radiation information from API's to the selected roofs. The inverter manager uses that information to predict feasibility.	
Methods	
public List<JSONObject> checkWeather(int inverterID)	Checks weather conditions on API's.
public List<JSONObject> checkSolarRadiationMap(int inverterID)	Checks solar radiation map to calculate feasibility.

Notification Manager	
This class handles notification services for users in an emergency situation. Moreover it provides instantaneous information about panels to the View Package.	
Methods	
public int sendUpdates(int userID, List<JSONObject> condition)	Send notification to the users on emergency conditions and about usage rate.

3.2.2 Data Package

Data	
The Facade class of the package. Every class in the package is a child class of Data class. It stores the creator user and the creation date.	
Attributes	
public User creator	This attribute points to the user responsible for the creation of this data. In case there is no such user it holds null pointer.
public Datatime date	This attribute holds the date and time when this data was created.
Methods	
Get and set methods	

User	
This class stores information about the users.	
Attributes	
private string email	This attribute points to the user responsible for the creation of this data. In case there is no such user it holds null pointer.
private string password	This attribute holds the date and time when this data was created.
private List<Building> buildings	This attribute points to all of the buildings the user owns.
private List<string> preferences	This attribute holds the user made preferences over the application to reload them whenever user logs in.
private currentLocation	This attribute holds the location user resides in.
private string name	This attribute holds the name of the user.
private string surname	This attribute holds the surname of the user.
Methods	
Get and set methods	

SolarPlan	
This class stores the user made solar plans. Used for the feasibility study of a building, it has no use for maintenance. It enables users to save their progress while creating a plan.	
Attributes	
private Building building	This attribute points to the building the solar plan is based on.

private List<Roof> roofs	This attribute holds the roofs solar panels are planned to be implemented on.
private List<SolarPanel> panels	This attribute holds the solar panels that are planned to be implemented.
private Inverter inverter	This attribute holds the inverter planned to be implemented.
Methods	
Get and set methods	

Building	
This class is responsible for holding information about buildings. Building class has roof objects because a single building may have several roofs.	
Attributes	
private string type	This attribute holds the type of the building.
private string name	This attribute holds the name of the building that is given by the owner user.
private string address	This attribute holds the address oft he building.
private int floorNo	This attribute holds the number of stories in the building.
private location	This attribute holds the location of the building over the map.
private List<Roof> roofs	This attribute points to the roofs of the building.
Methods	
Get and set methods	

BuildingEnergy

This class is used to store the past statistics about a building. It stores both how much electricity is bought from the grid and how much is sold.	
Attributes	
private Building building	This attribute points to the building this information is generated from.
private Datetime date	This attribute holds the starting point of this information.
private Time timeSpan	This attribute holds the total time this information is based on.
private double consumedEnergy	This attribute holds the amount of electricity consumed.
private double soldEnergy	This attribute holds the amount of electricity sold.
private double consumePrice	This attribute holds the ratio electricity bought for.
private double sellPrice	This attribute holds the ratio electricity is sold for.
Methods	
Get and set methods	

Roof	
This class contains all of the features of the roofs such as roof angle, sun angle, width, height, and surface area. These features will be used for parameters of calculating the feasibility score of the rooftops and estimated solar energy production.	
Attributes	
private double roofAngle	This attribute holds the angle of the roof.
private double surfaceArea	This attribute holds the surface area of the roof.

private double sunAngle	This attribute holds the angle of the sunlights coming to the roof.
private boolean isSuitable	This attribute holds the suitability of the roof.
private double width	This attribute holds the width of the roof.
private double height	This attribute holds the height of the roof.
private List<SolarPanel> panels	This attribute points to the solar panels over the roof.
Methods	
Get and set methods	

SolarPanel	
This class is used for identifying every solar panel registered in the application. It has some variables such as id, type, width, height, price, nominal power, warranty, efficiency coefficient, implementation, cost, and inverter type which will be used for the maintenance service.	
Attributes	
private string type	This attribute holds the type of the panel.
private double width	This attribute holds the width of the panel.
private double height	This attribute holds the height of the panel.
private int price	This attribute holds the price of the panel.
private double nominalPower	This attribute holds the nominal power of the panel.
private int warranty	This attribute holds the warranty of the panel.
private double efficiencyCoefficient	This attribute holds the efficiency

	coefficient of the panel. The older the panel the more it degrades, the less efficient the panel is.
private double implementationCost	This attribute holds the implementation cost of the panel.
private Inverter inverter	This attribute points to the inverter connected to the panel.
Methods	
Get and set methods	

EnergyProduction	
This class is used to store the past statistics of generated electricity for panels.	
Attributes	
private SolarPanel panel	This attribute points to the panel this information is generated for.
private double producedEnergy	This attribute holds the total amount of energy produced
private Datetime date	This attribute holds the starting point of this information.
private Time timeSpan	This attribute holds the total time this information is based on.
Methods	
Get and set methods	

Inverter	
This class holds general information and properties of an inverter.	
Attributes	
private string type	This attribute holds the type of the inverter.

private URL generatedEnergyPointer	This attribute points to the connection for generated electricity information.
Methods	
Get and set methods	

4. 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 modelling and visualization of components in softwares.

5. 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].