**Languages/Tools:-**

1. **IDE:**
   1. Eclipse
2. **Programming Language:**
   1. Backend
      1. Java
   2. Frontend
      1. HTML
      2. CSS
      3. JavaScript
3. **Framework:**
   1. Java EE (Java Servlet)
4. **Packages/Libraries:**
   1. Backend
      1. Gson: To convert object to JSON
   2. Frontend
      1. Bootstrap: To design the UI
      2. AnyChart: To plot the graph on the web page
5. **Server:**
   1. Apache Tomcat

**CSV Files:**-  
Here, we used two CSV files. One CSV files contains the solar radiation in each month. This CSV file has three columns. The first columns has name of the months, and the second column has the monthly avg. solar radiation (in kW/m2).

Similarly, the second CSV files contains the average wind speed in each month. This CSV file has two columns. The first columns has name of the months and second column has the average wind speed of the corresponding months.

**Formulas:-**

* **Solar Energy Calculation:**

Energy generated (in kW) by solar panels in a particular month is calculated as follows:

Performance ratio or coefficient for losses range between 0.5 and 0.9.

* **Wind Energy Calculation:**

Energy generated (in kW) by a wind turbine in a particular month is calculated as follows:

Density of the air and Length of the blades is constant. The standard density of air is 1.225 kg/m3.

**Algorithm:**

**Calculate Solar Electricity Generated**

1. Input: Receive user inputs for the number of solar panels (solarTotal), area of each solar panel (solarArea) solar panel efficiency (solarEfficiency), performance ratio (solarPerformence), and the uploaded CSV file containing monthly solar radiation.
2. Read CSV File: Parse the uploaded CSV file to extract the month and the corresponding solar radiations for each month.
3. Calculate Electricity Generated: For each month, calculate the electricity generated using the formula:

electricityGenerated = solarTotal \* solarArea \* (solarEfficiency / 100) \* solarPerformence \* solarRadiation

1. Store Calculated Data: Store the calculated electricity generated values and the corresponding months in lists (electricityGeneratedList and monthsList).
2. Scale Data: Find the maximum value in electricityGeneratedList for scaling the graph.
3. Prepare Response: Package the electricityGeneratedList and monthsList into a map (responseData).
4. Send Response: Convert the responseData map to JSON format and send it as the response to the client.
5. Handle Errors: Implement error handling to manage any exceptions that may occur during the process, such as invalid user inputs or file format issues.
6. Display Graph: Use JavaScript to parse the JSON response and plot a bar graph showing the electricity generated for each month.
7. Output: Display the graph to the user on the webpage.

**Calculate Wind Electricity Generated**

1. Input: Receive user inputs for the number of wind turbine (windTotal), density of the air (airDensity), length of the blades (bladeLength) and the uploaded CSV file containing monthly wind speed.
2. Read CSV File: Parse the uploaded CSV file to extract the month and the corresponding wind speed for each month.
3. Calculate Electricity Generated: For each month, calculate the electricity generated using the formula:

electricityGenerated = (0.5 \* rho \* 3.14159 \* Math.pow(R, 2) \* Math.pow(speed, 3) \* N) / 1000

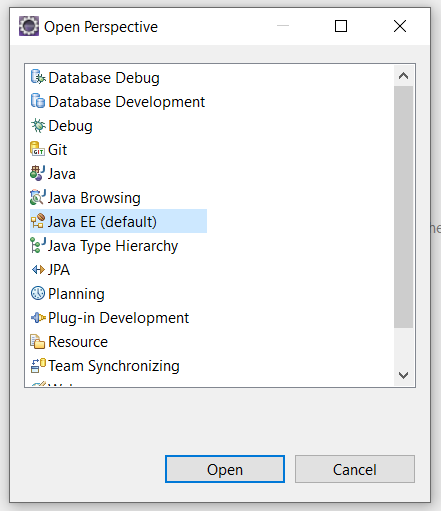
1. Store Calculated Data: Store the calculated electricity generated values and the corresponding months in lists (electricityGeneratedList and monthsList).
2. Scale Data: Find the maximum value in electricityGeneratedList for scaling the graph.
3. Prepare Response: Package the electricityGeneratedList and monthsList into a map (responseData).
4. Send Response: Convert the responseData map to JSON format and send it as the response to the client.
5. Handle Errors: Implement error handling to manage any exceptions that may occur during the process, such as invalid user inputs or file format issues.
6. Display Graph: Use JavaScript to parse the JSON response and plot a bar graph showing the electricity generated for each month.
7. Output: Display the graph to the user on the webpage.

**Procedures:-**

**1.** Download and install the **Eclipse IDE for Enterprise Java and Web Developers** from the link <https://www.eclipse.org/downloads/packages/release/2024-03/r/eclipse-ide-enterprise-java-and-web-developers>

IDE stands for Integrated Development Environment. It's a software application that provides comprehensive facilities to computer programmers for software development. An IDE typically consists of a source code editor, build automation tools, debugger, and other features that streamline the process of writing, testing, and debugging code.

**2.** Open the installed IDE and at the top-right side in the IDE, there is an icon  (just after the search icon) to open the *Perspective.* Click on that icon and then a dialog box (shown in below image) will appear, on the appeared dialog box select *JAVA EE* and the click ***Open***.

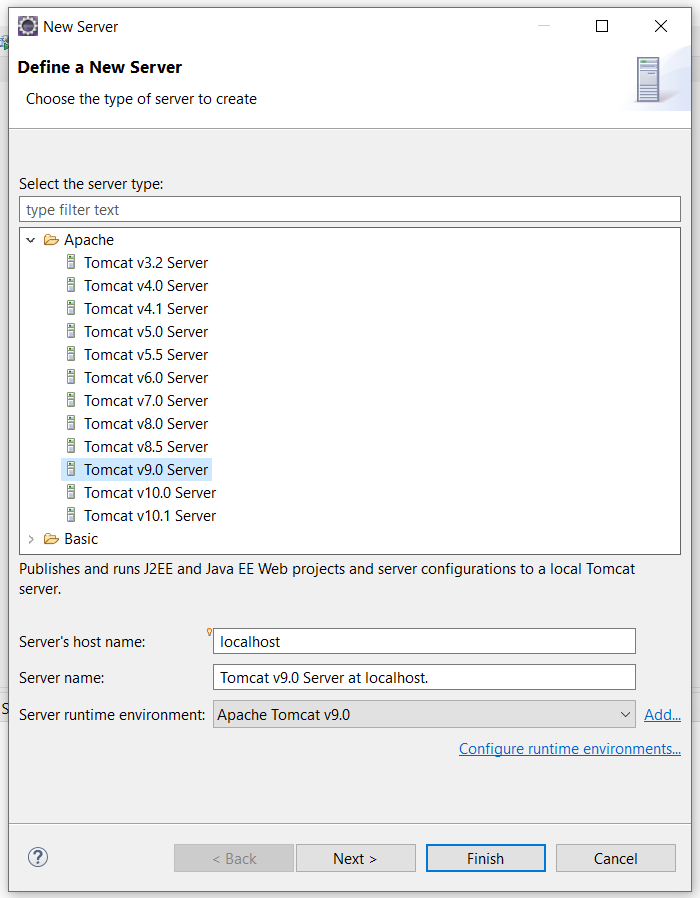


**3.** Now, download the **Apache Tomcat server** (Core Binary Distribution) from the link <https://tomcat.apache.org/download-90.cgi>. After downloaded the zip or tar.gz file, extract/un-zip the file to get the folder.

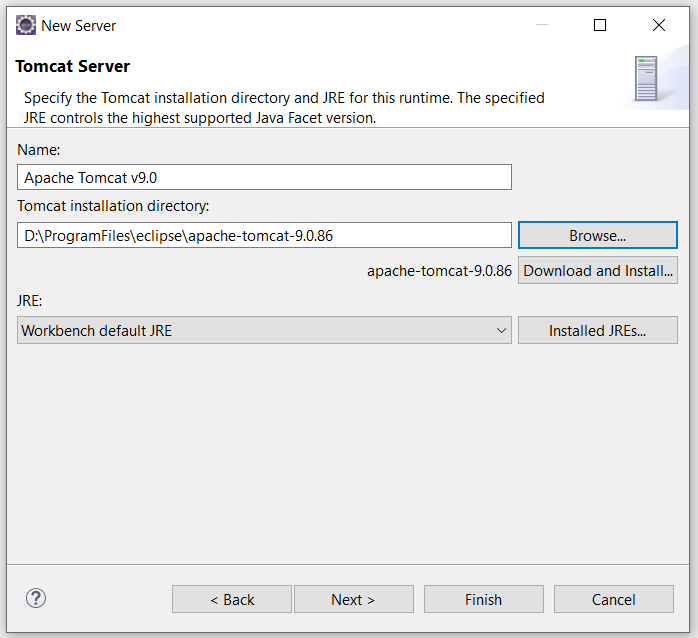
Apache Tomcat is an open-source web server and servlet container developed by the Apache Software Foundation. It implements several Java EE (Enterprise Edition) specifications, including Java Servlet, JavaServer Pages (JSP), WebSocket, and Java Expression Language (EL). Tomcat provides a servlet container that manages the execution of Java servlets and JavaServer Pages (JSP), allowing developers to create dynamic web applications using Java technologies.

**4.** On the Eclipse IDE, at the bottom you will see a panel. In this panel, click on the servers tab. In this tab it will display a text “No servers are available. Click this link to create a new server”. Click on this text and a dialog box will open (shown in the below image).

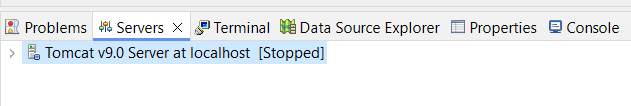
Now, click the **Apache** folder to expend it, and then select the ***Tomcat v9.0 Server*** from the list. And click the ***Next*** button.



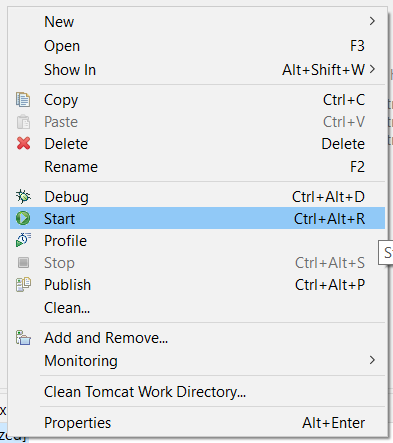
Now click the ***Browse*** button and select the folder extracted in step 3. And then click the ***Finish*** button.



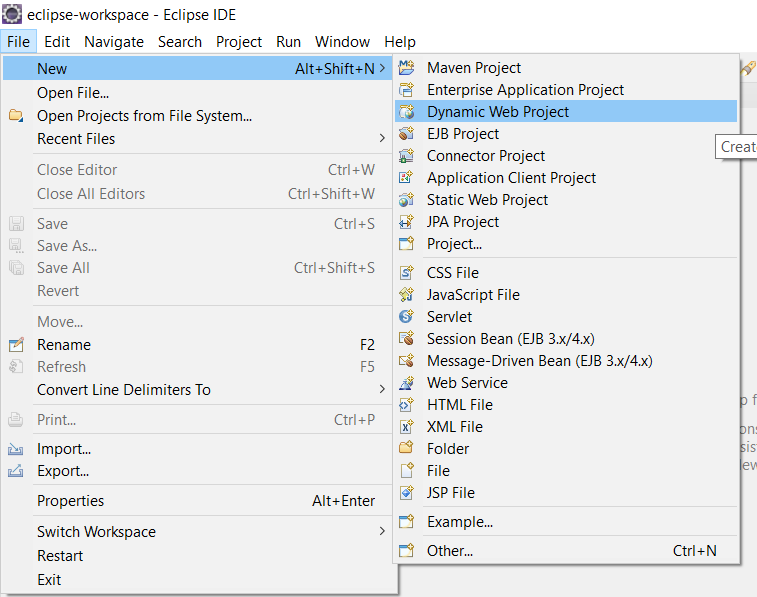
Now the Tomcat server will show in **Servers** tab (shown in the below image).



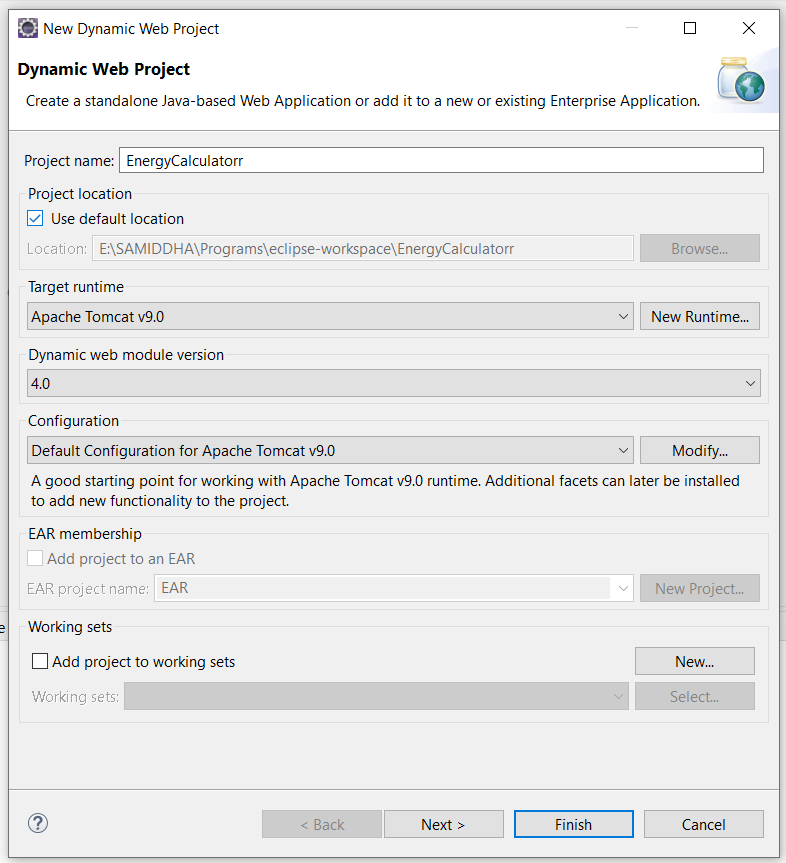
But the server is Stopped. To start the server, do right-click on the server and then select **Start.**

****

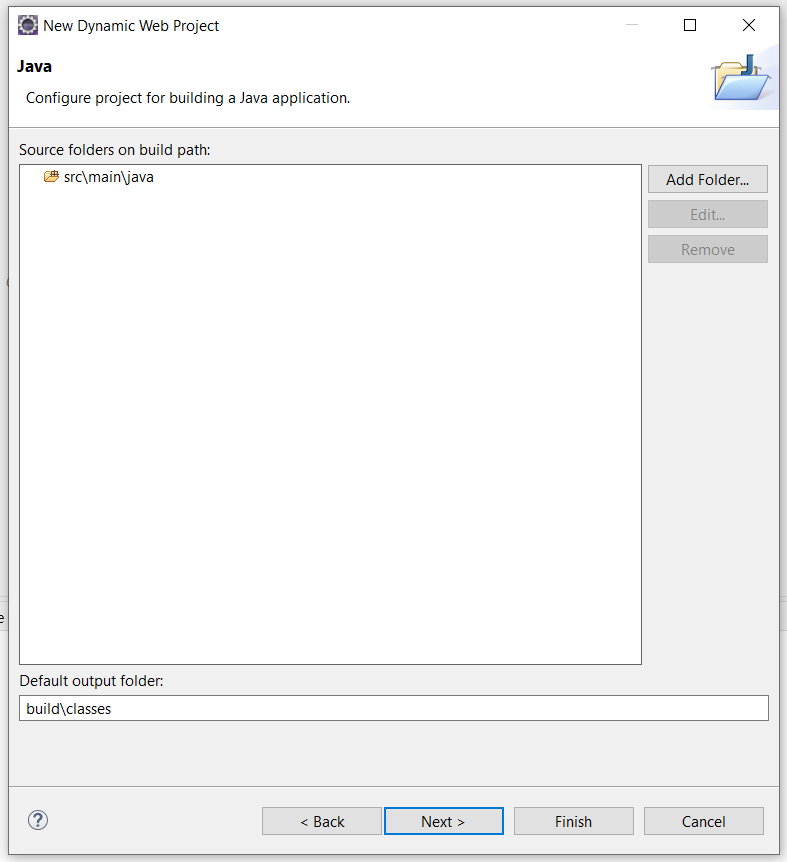
**5.** Now on the top-left side go to **File 🡪 New 🡪 Dynamic Web Project** (shown in below image).



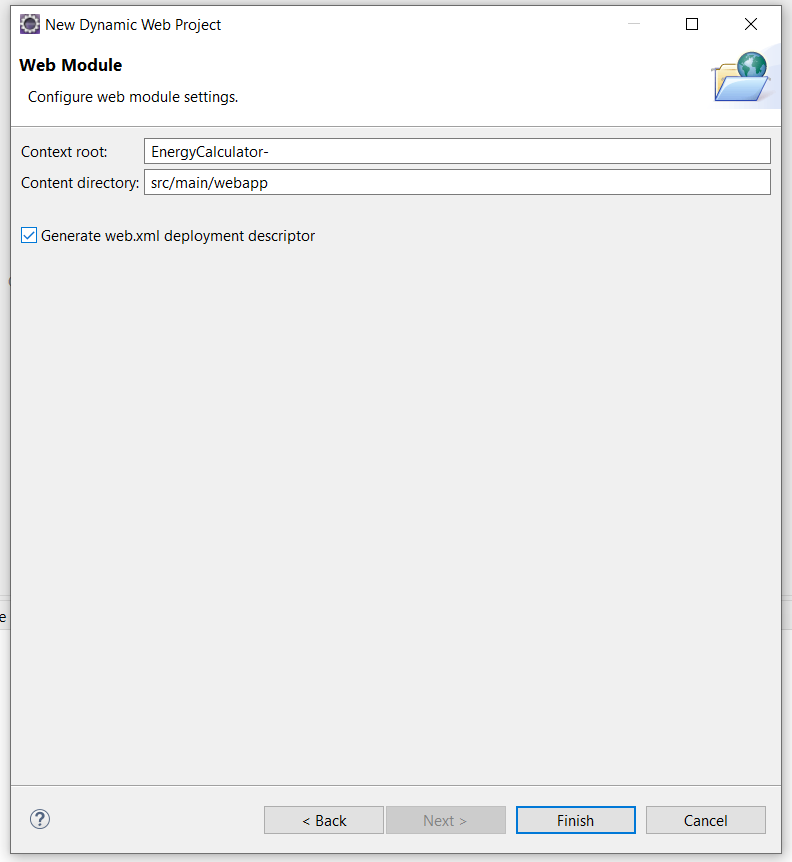
After clicking **Dynamic Web Project**, a dialog box will open (shown in below image). In the *Project Name* field enter a name for the project. Here, the entered project name is EnergyCalculator And click **Next.**



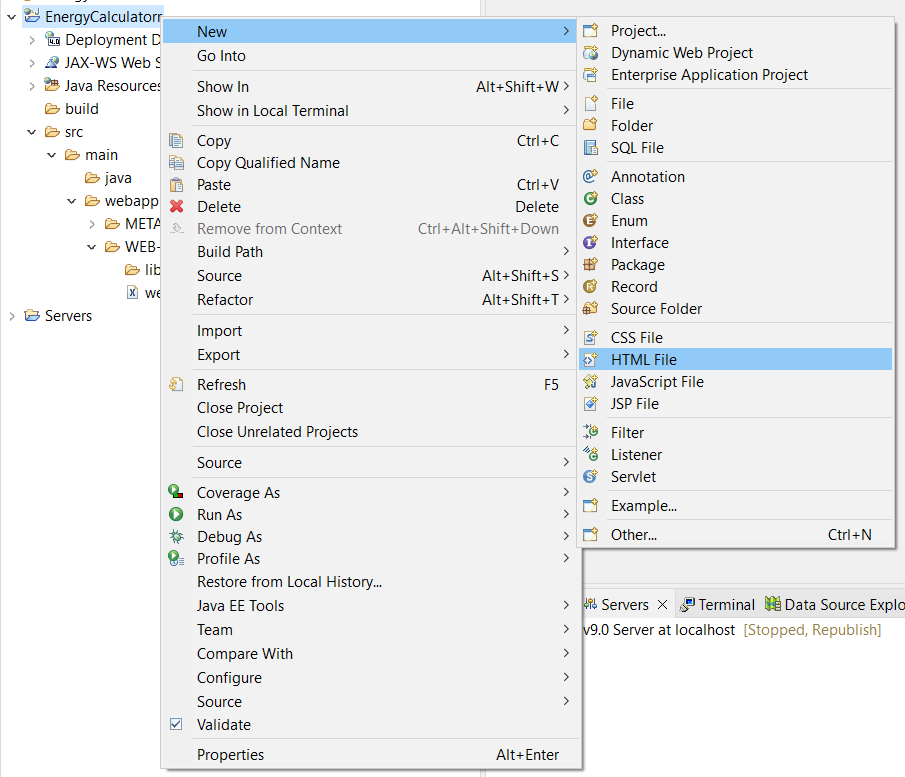
Now another screen (shown in below image) will be open. On that screen just click the **Next** button again.

****

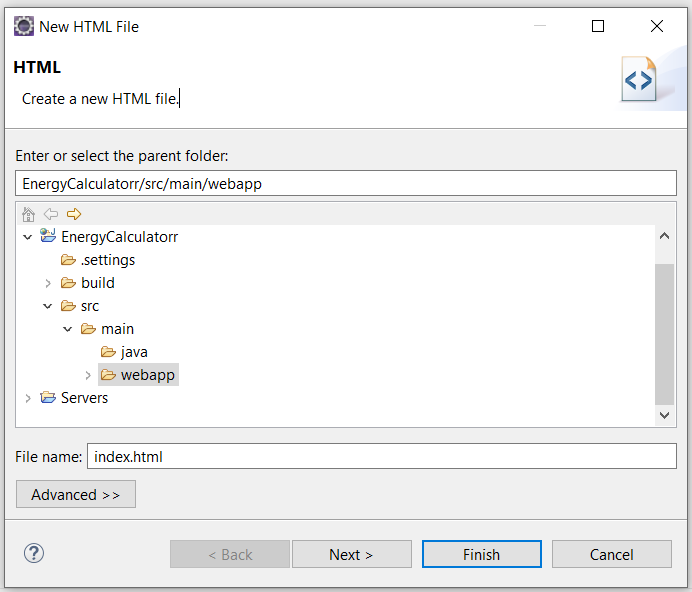
Now on the next screen (shown in below image), click the checkbox (Generate web.xml), and click the **Finish** button.



**6.** In the left panel, from the **Project Explorer,** right-click on the created project (EnergyCalculator), then click **New 🡪 HTML File** (shown in below image).



Now, name the HTML file *to index.html* and click on the **Finish** button.



Paste the below code to the created **index.html** file.

<!**DOCTYPE** html>

<**html**>

<**head**>

<**title**>Energy Calculator</**title**>

<**meta** name=*"viewport"* content=*"width=device-width, initial-scale=1, shrink-to-fit=yes"*>

<**script** src=*"https://code.jquery.com/jquery-3.7.1.min.js"* integrity=*"sha256-/JqT3SQfawRcv/BIHPThkBvs0OEvtFFmqPF/lYI/Cxo="* crossorigin=*"anonymous"*></**script**>

<**script** src=*"https://cdn.anychart.com/releases/8.12.0/js/anychart-bundle.min.js"*></**script**>

<**link** rel=*"stylesheet"* href=*"https://cdn.anychart.com/releases/8.12.0/css/anychart-ui.min.css"* type=*"text/css"*>

<**link** rel=*"stylesheet"* type=*"text/css"* href=*"https://cdn.anychart.com/releases/8.12.0/fonts/css/anychart-font.min.css"*/>

<**link** href=*"https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"* rel=*"stylesheet"* integrity=*"sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"* crossorigin=*"anonymous"*>

<**script** src=*"https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"* integrity=*"sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"* crossorigin=*"anonymous"*></**script**>

<**script** src=*"assets/js/fontawesome6.5.1.js"*></**script**>

<**style**>

body{

overflow-x:*hidden*;

padding: *10px*;

background-image: *url('assets/images/background.jpg')*;

background-position: *center*;

background-repeat: *no-repeat*;

background-size: *cover*;

background-attachment: *fixed*;

}

.anychart-credits{

display: *none !important*;

}

.container-graph{

text-align: *center !important*;

height: *480px*;

margin: *20px 0px 10px 0px*;

padding: *0*;

box-shadow: *0px 0px 5px 1px gray*;

-webkit-box-shadow: *0px 0px 5px 1px gray*;

-moz-box-shadow: *0px 0px 5px 1px gray*;

/\* position: relative;

bottom: 5px; \*/

}

#windGraph{

position: *relative*;

bottom: *-55px*;

}

.solarForm-input, .windForm-input{

outline: *none !important*;

border-radius: *5px*;

border: *1px solid #00e01e*;

color: *#01b728 !important*;

}

.solarForm-input:focus, .windForm-input:focus{

border: *1px solid #8dfb9c*;

box-shadow: *0px 0px 1px 1px #02f723*;

-webkit-box-shadow: *0px 0px 1px 1px #02f723*;

-moz-box-shadow: *0px 0px 1px 1px #02f723*;

}

.solarForm-input-group-text, .windForm-input-group-text{

border: *1px solid #00e01e*;

background-color: *#00d62e*;

color: *white*;

}

.heading{

color: *rgb(0 209 159)*;

font-size: *25px*;

margin-bottom: *15px*;

font-weight: *bold*

}

.solarForm-label, .windForm-label{

color: *#515151*;

}

.source-link{

color: *white*;

text-decoration: *none !important*;

}

.source-link:hover{

color:*#ceebfd*

}

.input-HelpBlock{

color: *black*;

}

/\* @media only screen and (min-device-width : 768px){

#windGraph{

margin-top: 20px;

}

} \*/

</**style**>

</**head**>

<**body**>

<**div** class=*"row"* style="gap:*15px*">

<**div** class=*"col-md mb-5"*>

<**div** class=*"heading"*>Calculate Electricity Generated by Solar Energy</**div**>

<**form** id=*"solarForm"* action=*"/EnergyCalculator/calculate-solar-energy"* method=*"post"* enctype=*"multipart/form-data"*>

<**div** class=*"row mb-3"*>

<**div** class=*"col-md text-md-end"*>

<**label** class=*"solarForm-label"* for=*"solarTotal"*><**b**>Total No. of Solar Panel:</**b**></**label**>

</**div**>

<**div** class=*"col-md"*>

<**div** class=*"input-group"*>

<**input** type=*"number"* step=*"1"* class=*"form-control solarForm-input"* placeholder=*"Total Solar Panels"* id=*"solarTotal"* name=*"solarTotal"* required form=*"solarForm"*>

<**span** class=*"input-group-text solarForm-input-group-text"*>Unit</**span**>

</**div**>

</**div**>

</**div**>

<**div** class=*"row mb-3"*>

<**div** class=*"col-md text-md-end"*>

<**label** class=*"solarForm-label"* for=*"solarArea"*><**b**>Area of the Solar Panel (in m<**sup**>2</**sup**>):</**b**></**label**>

</**div**>

<**div** class=*"col-md"*>

<**div** class=*"input-group"*>

<**input** type=*"number"* step=*"0.01"* class=*"form-control solarForm-input"* placeholder=*"Panel Area"* id=*"solarArea"* name=*"solarArea"* required form=*"solarForm"*>

<**span** class=*"input-group-text solarForm-input-group-text"*>m<**sup**>2</**sup**></**span**>

</**div**>

</**div**>

</**div**>

<**div** class=*"row mb-3"*>

<**div** class=*"col-md text-md-end"*>

<**label** class=*"solarForm-label"* for=*"solarEfficiency"*><**b**>Efficiency or yield of the Solar Panel (%):</**b**>&nbsp;<**i** class=*"fa-solid fa-circle-info"* data-bs-content=*"Yield of solar panel = electrical power (in kWp) of one solar panel divided by the area of one panel"* data-bs-container=*"body"* data-bs-toggle=*"popover"* data-bs-placement=*"top"* data-bs-trigger=*"hover"* data-bs-html=*"false"*></**i**></**label**>

</**div**>

<**div** class=*"col-md"*>

<**div** class=*"input-group"*>

<**input** type=*"number"* step=*"0.01"* class=*"form-control solarForm-input"* placeholder=*"Efficiency"* id=*"solarEfficiency"* name=*"solarEfficiency"* required form=*"solarForm"*>

<**span** class=*"input-group-text solarForm-input-group-text"*>%</**span**>

</**div**>

</**div**>

</**div**>

<**div** class=*"row mb-3"*>

<**div** class=*"col-md text-md-end"*>

<**label** class=*"solarForm-label"* for=*"solarPerformance"*><**b**>Performance Ratio:</**b**>&nbsp;<**i** class=*"fa-solid fa-circle-info"* data-bs-content=*"Performance ratio or coefficient for losses generally ranges between 0.5 to 0.9<br><b>Example of losses that gives the PR value:</b><ul><li>shadow losses</li><li>Temperature losses</li><li>Losses due to dust</li><li>DC cables losses</li><li>AC cables losses</li><li>Inverter losses</li></ul>"* data-bs-container=*"body"* data-bs-toggle=*"popover"* data-bs-placement=*"top"* data-bs-trigger=*"hover"* data-bs-html=*"true"*></**i**></**label**>

</**div**>

<**div** class=*"col-md"*>

<**div** class=*"input-group"*>

<**input** type=*"number"* step=*"0.1"* class=*"form-control solarForm-input"* placeholder=*"Performance Ratio"* id=*"solarPerformance"* name=*"solarPerformance"* required form=*"solarForm"*>

<!-- <span class="input-group-text solarForm-input-group-text"></span> -->

</**div**>

</**div**>

</**div**>

<**div** class=*"row mb-3"*>

<**div** class=*"col-md text-md-end"*>

<**label** class=*"solarForm-label"* for=*"solarCSV"*><**b**>Monthly Avg. Solar Radiation:</**b**></**label**>

</**div**>

<**div** class=*"col-md"*>

<**div** class=*"input-group"*>

<**input** type=*"file"* class=*"form-control solarForm-input"* id=*"solarCSV"* name=*"solarCSV"* accept=*".csv"* required form=*"solarForm"*>

<**label** class=*"input-group-text solarForm-input-group-text"* for=*"solarCSV"*>KW/m<**sup**>2</**sup**></**label**>

</**div**>

<**div** id=*"solarCSVHelpBlock"* class=*"form-text input-HelpBlock"*>

Upload the CSV file to read the data of avg. hour of sunlight of each month

</**div**>

</**div**>

</**div**>

<**div** class=*"row"*>

<**div** class=*"col text-end"*>

<**button** class=*"btn btn-danger"* id=*"solarForm-reset"* type=*"reset"* form=*"solarForm"*>Reset</**button**>

<**button** class=*"btn btn-primary"* id=*"solarForm-submit"* type=*"submit"* form=*"solarForm"*>Calculate</**button**>

</**div**>

</**div**>

</**form**>

<**div** class=*"container-graph"* id=*"solarGraph"*></**div**>

</**div**>

<**div** class=*"col-md mb-5"*>

<**div** class=*"heading"*>Calculate Electricity Generated by Wind Energy</**div**>

<**form** id=*"windForm"* action=*"/EnergyCalculator/calculate-wind-energy"* method=*"post"* enctype=*"multipart/form-data"*>

<**div** class=*"row mb-3"*>

<**div** class=*"col-md text-md-end"*>

<**label** class=*"windForm-label"* for=*"windTotal"*><**b**>Total No. of Wind Turbines:</**b**></**label**>

</**div**>

<**div** class=*"col-md"*>

<**div** class=*"input-group"*>

<**input** type=*"number"* step=*"1"* class=*"form-control windForm-input"* placeholder=*"Total Wind Turbines"* id=*"windTotal"* name=*"windTotal"* required form=*"windForm"*>

<**span** class=*"input-group-text windForm-input-group-text"*>Unit</**span**>

</**div**>

</**div**>

</**div**>

<**div** class=*"row mb-3"*>

<**div** class=*"col-md text-md-end"*>

<**label** class=*"windForm-label"* for=*"airDensity"*><**b**>Air Density (in kg/m<**sup**>3</**sup**>):</**b**>&nbsp;<**i** class=*"fa-solid fa-circle-info"* data-bs-content=*'<div>The standard density of the air is 1.225 kg/m<sup>3</sup></div>'* data-bs-container=*"body"* data-bs-toggle=*"popover"* data-bs-placement=*"top"* data-bs-trigger=*"hover"* data-bs-html=*"true"*></**i**></**label**>

</**div**>

<**div** class=*"col-md"*>

<**div** class=*"input-group"*>

<**input** type=*"number"* step=*"0.01"* class=*"form-control windForm-input"* placeholder=*"Density"* id=*"airDensity"* name=*"airDensity"* required form=*"windForm"*>

<**span** class=*"input-group-text windForm-input-group-text"*>kg/m<**sup**>3</**sup**></**span**>

</**div**>

</**div**>

</**div**>

<**div** class=*"row mb-3"*>

<**div** class=*"col-md text-md-end"*>

<**label** class=*"windForm-label"* for=*"bladeLength"*><**b**>Length of the Blades (in Meters):</**b**></**label**>

</**div**>

<**div** class=*"col-md"*>

<**div** class=*"input-group"*>

<**input** type=*"number"* step=*"0.1"* class=*"form-control windForm-input"* placeholder=*"Length"* id=*"bladeLength"* name=*"bladeLength"* required form=*"windForm"*>

<**span** class=*"input-group-text windForm-input-group-text"*>M</**span**>

</**div**>

</**div**>

</**div**>

<**div** class=*"row mb-3"*>

<**div** class=*"col-md text-md-end"*>

<**label** class=*"windForm-label"* for=*"windCSV"*><**b**>Monthly avg. Wind Speed:</**b**></**label**>

</**div**>

<**div** class=*"col-md"*>

<**div** class=*"input-group"*>

<**input** type=*"file"* class=*"form-control windForm-input"* id=*"windCSV"* name=*"windCSV"* accept=*".csv"* required form=*"windForm"*>

<**label** class=*"input-group-text windForm-input-group-text"* for=*"windCSV"*>m/s</**label**>

</**div**>

<**div** id=*"windCSVHelpBlock"* class=*"form-text input-HelpBlock"*>

Upload the CSV file to read the data of avg. wind speed of each month

</**div**>

</**div**>

</**div**>

<**div** class=*"row"*>

<**div** class=*"col text-end"*>

<**button** class=*"btn btn-danger"* id=*"windForm-reset"* type=*"reset"* form=*"windForm"*>Reset</**button**>

<**button** class=*"btn btn-primary"* id=*"windForm-submit"* type=*"submit"* form=*"windForm"*>Calculate</**button**>

</**div**>

</**div**>

</**form**>

<**div** class=*"container-graph"* id=*"windGraph"*></**div**>

</**div**>

</**div**>

<**h2**>Sources:</**h2**>

<**ul**>

<**li** style="color:*white*"><**a** class=*"source-link"* href=*"https://www.linkedin.com/pulse/how-calculate-output-solar-pv-system-detailed-guide-eman-rohayem/"* target=*"\_blank"*>How to Calculate the Output of a Solar PV System</**a**></**li**>

<**li** style="color:*white*"><**a** class=*"source-link"* href=*"https://photovoltaic-software.com/principle-ressources/how-calculate-solar-energy-power-pv-systems"* target=*"\_blank"*>How to calculate the annual solar energy output of a photovoltaic system?</**a**></**li**>

<**li** style="color:*white*"><**a** class=*"source-link"* href=*"https://thundersaidenergy.com/downloads/wind-power-impacts-of-larger-turbines/"* target=*"\_blank"*>Windy physics: how is power of a wind turbine calculated?</**a**></**li**>

<**li** style="color:*white*"><**a** class=*"source-link"* href=*"https://www.e-education.psu.edu/emsc297/node/649"* target=*"\_blank"*>Wind Energy and Power Calculations</**a**></**li**>

</**ul**>

<**script**>

**function** plotSolarGraph(data=[]){

chart\_solar = anychart.area();

// enable animation and set animation duration to 2 sec

chart\_solar.animation(**true**, 2000);

// set the chart title

**var** title = chart\_solar.title();

title.enabled(**true**);

title.useHtml(**true**);

title.text("Monthly Electricity Generated by Solar Energy");

title.padding(13, 0, 5, 0)

//title.fontSize(12);

// set the titles of the axes

chart\_solar.xAxis().title("Month");

chart\_solar.yAxis().title("Electricity Generated");

chart\_solar.xAxis().title().fontColor("gray");

chart\_solar.yAxis().title().fontColor("gray");

// set the orientation of the axis label

chart\_solar.yAxis().orientation("left");

chart\_solar.xAxis().orientation("bottom");

// adjusting axes labels

chart\_solar.yAxis().labels().fontSize(11);

chart\_solar.yAxis().labels().rotation(-90);

chart\_solar.yAxis().labels().padding(0,20,0,0);

chart\_solar.xAxis().labels().fontSize(11);

chart\_solar.xAxis().labels().rotation(-45);

chart\_solar.xAxis().labels().padding(0,0,0,0);

//configure tooltip text

chart\_solar.tooltip().format(`Energy: {%value}`);

**var** credits = chart\_solar.credits();

credits.enabled(**false**);

**if**(data.length <= 0){

noDataLabel = chart\_solar.noData().label();

noDataLabel.enabled(**true**);

noDataLabel.text("No Data");

noDataLabel.background().enabled(**true**);

noDataLabel.background().fill("White 1");

noDataLabel.background().stroke("2 gray");

noDataLabel.padding(20);

noDataLabel.fontSize(30);

noDataLabel.fontColor("gray");

noDataLabel.fontVariant("small-caps");

}

**else**{

// enable the crosshair

chart\_solar.crosshair(**true**);

// enable major grids

chart\_solar.xGrid().enabled(**true**);

chart\_solar.yGrid().enabled(**true**);

// enable minor grids

chart\_solar.xMinorGrid().enabled(**true**);

chart\_solar.yMinorGrid().enabled(**true**);

**var** series = chart\_solar.line(data);

}

// enable context menu

chart\_solar.contextMenu(**true**);

// set container and draw chart

$("#solarGraph").html('');

chart\_solar.container("solarGraph");

chart\_solar.draw();

}

$("#solarForm").on('reset', **function**(){

plotSolarGraph();

});

$("#solarForm").on('submit', **function**(event){

event.preventDefault();

**var** theForm = **new** FormData(**this**);

$("#solarForm-submit").prop("disabled", **true**);

$("#solarForm-submit").html("Calculating");

$.ajax(

{

type: 'POST',

url: $(**this**).attr('action'),

data: theForm,

contentType: **false**,

processData: **false**,

//dataType: 'json',

cache: **false**,

success: **function** (response) {

**var** data = [];

**for**(**var** i=0; i<response.monthsList.length; i++){

**var** d = {

x: response.monthsList[i],

value: response.electricityGeneratedList[i]

}

data.push([d['x'], d['value']]);

}

plotSolarGraph(data);

$("#solarForm-submit").prop("disabled", **false**);

$("#solarForm-submit").html("Calculate");

},

error: **function** (response) {

$("#solarForm-submit").prop("disabled", **false**);

$("#solarForm-submit").html("Calculate");

alert("Something went wrong!");

}

});

});

**function** plotWindGraph(data=[]){

chart\_wind = anychart.line();

// enable animation and set animation duration to 2 sec

chart\_wind.animation(**true**, 2000);

// set the chart title

**var** title = chart\_wind.title();

title.enabled(**true**);

title.useHtml(**true**);

title.text("Monthly Electricity Generated by Wind Energy");

title.padding(13, 0, 5, 0)

//title.fontSize(12);

// set the titles of the axes

chart\_wind.xAxis().title("Month");

chart\_wind.yAxis().title("Electricity Generated");

chart\_wind.xAxis().title().fontColor("gray");

chart\_wind.yAxis().title().fontColor("gray");

// set the orientation of the axis label

chart\_wind.yAxis().orientation("left");

chart\_wind.xAxis().orientation("bottom");

// adjusting axes labels

chart\_wind.yAxis().labels().fontSize(11);

chart\_wind.yAxis().labels().rotation(-90);

chart\_wind.yAxis().labels().padding(0,20,0,0);

chart\_wind.xAxis().labels().fontSize(11);

chart\_wind.xAxis().labels().rotation(-45);

chart\_wind.xAxis().labels().padding(0,0,0,0);

//configure tooltip text

chart\_wind.tooltip().format(`Energy: {%value}`);

**var** credits = chart\_wind.credits();

credits.enabled(**false**);

**if**(data.length <= 0){

noDataLabel = chart\_wind.noData().label();

noDataLabel.enabled(**true**);

noDataLabel.text("No Data");

noDataLabel.background().enabled(**true**);

noDataLabel.background().fill("White 1");

noDataLabel.background().stroke("2 gray");

noDataLabel.padding(20);

noDataLabel.fontSize(30);

noDataLabel.fontColor("gray");

noDataLabel.fontVariant("small-caps");

}

**else**{

// enable the crosshair

chart\_wind.crosshair(**true**);

// enable major grids

chart\_wind.xGrid().enabled(**true**);

chart\_wind.yGrid().enabled(**true**);

// enable minor grids

chart\_wind.xMinorGrid().enabled(**true**);

chart\_wind.yMinorGrid().enabled(**true**);

**var** series = chart\_wind.line(data);

}

// enable context menu

chart\_wind.contextMenu(**true**);

// set container and draw chart

$("#windGraph").html('');

chart\_wind.container("windGraph");

chart\_wind.draw();

}

$("#windForm").on('reset', **function**(){

plotWindGraph();

});

$("#windForm").on('submit', **function**(event){

event.preventDefault();

**var** theForm = **new** FormData(**this**);

$("#windForm-submit").prop("disabled", **true**);

$("#windForm-submit").html("Calculating");

$.ajax(

{

type: 'POST',

url: $(**this**).attr('action'),

data: theForm,

contentType: **false**,

processData: **false**,

//dataType: 'json',

cache: **false**,

success: **function** (response) {

**var** data = [];

**for**(**var** i=0; i<response.monthsList.length; i++){

**var** d = {

x: response.monthsList[i],

value: response.electricityGeneratedList[i]

}

data.push(d);

}

plotWindGraph(data);

$("#windForm-submit").prop("disabled", **false**);

$("#windForm-submit").html("Calculate");

},

error: **function** (response) {

$("#windForm-submit").prop("disabled", **false**);

$("#windForm-submit").html("Calculate");

alert("Something went wrong!");

}

});

});

$(window).on('load', **function**(){

$('[data-bs-toggle="popover"]').popover();

plotSolarGraph()

plotWindGraph();

})

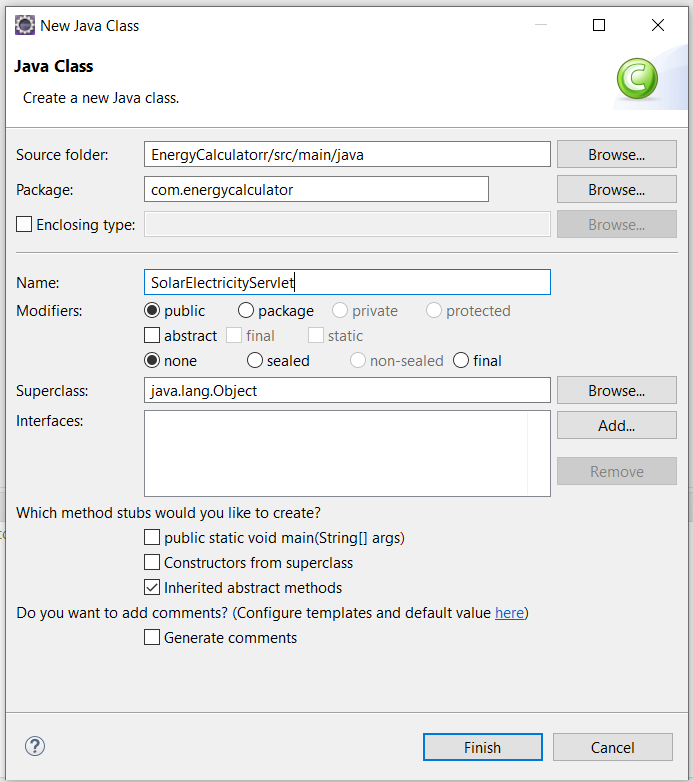
</**script**>

</**body**>

</**html**>

**7.** Now again, in the left panel, from the **Project Explorer,** right-click on the created project (EnergyCalculator), then click on **New 🡪 Class** to create .java files. After click on **Class,** a dialog box will open (shown in below image).

We will create four java files – 1) SolarElectricityCalculatorAndGraph.java, 2) SolarElectricityServlet.java, 3) WindElectricityCalculatorAndGraph.java, 4) WindElectricityServlet.java. So, perform this step four times to create each java file.



In *Package* field, enter the package name as ‘com.energycalculator’ for all the four java files. And in *Name* field enter the name of the java file in each case. Finally, all the four java files are created inside the folder ***src/main/java/com/energycalculator***.

Now, paste the below code to the created java file:

**I. SolarElectricityCalculatorAndGraph.java**

package com.energycalculator;

import javax.swing.\*;

import java.awt.\*;

import java.awt.geom.\*;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.\*;

import javax.servlet.http.\*;

import java.io.\*;

public class SolarElectricityCalculatorAndGraph extends JPanel {

public List<Double> electricityGeneratedList = new ArrayList<>();

public List<String> monthsList = new ArrayList<>();

public double maxVal;

int solarTotal;

double solarArea;

double solarEfficiency;

double solarPerformance;

InputStream csvFile; // Path to the CSV file containing solar hours data

public SolarElectricityCalculatorAndGraph(int total\_panel, double solar\_Area, double solar\_Efficiency, double solar\_Performance, InputStream csv\_File) {

this.solarTotal = total\_panel;

this.solarArea = solar\_Area;

this.solarEfficiency = solar\_Efficiency;

this.solarPerformance = solar\_Performance;

this.csvFile = csv\_File;

readDataFromFile(csvFile);

}

private void readDataFromFile(InputStream csv\_file) {

String line; // String variable to store each line read from the CSV file

String cvsSplitBy = ","; // CSV delimiter, assuming the CSV file is comma-separated

try (BufferedReader br = new BufferedReader(new InputStreamReader(csv\_file))) {

// Skip the first line (header) as it contains column names

br.readLine();

// Read each line of the CSV file and extract the Solar Electricity data

while ((line = br.readLine()) != null) {

String[] data = line.split(cvsSplitBy);

String month = data[0]; // Assuming month is in column 1

double solarRadiations = Double.*parseDouble*(data[1]); // Assuming solar radiations are in column 2

double electricityGenerated = solarTotal \* solarArea \* (solarEfficiency / 100) \* solarPerformance \* solarRadiations;

// Add data to the lists

monthsList.add(month);

electricityGeneratedList.add(electricityGenerated);

}

} catch (IOException e) {

e.printStackTrace();

}

// Find max value for scaling

maxVal = electricityGeneratedList.stream().mapToDouble(Double::doubleValue).max().orElse(0);

}

public static void main(String[] args) {

}

}

**II. SolarElectricityServlet.java**

package com.energycalculator;

import javax.servlet.\*;

import javax.servlet.annotation.\*;

import javax.servlet.http.\*;

import java.io.\*;

import java.util.\*;

//import org.json.JSONObject;

import com.google.gson.Gson;

import com.energycalculator.SolarElectricityCalculatorAndGraph;

*@WebServlet*(name="SolarElectricityServlet", urlPatterns={"/calculate-solar-energy/"})

*@MultipartConfig*

public class SolarElectricityServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

try {

// Get user input from the HTML form

int solarTotal = Integer.*parseInt*(request.getParameter("solarTotal"));

double solarArea = Double.*parseDouble*(request.getParameter("solarArea"));

double solarEfficiency = Double.*parseDouble*(request.getParameter("solarEfficiency"));

double solarPerformance = Double.*parseDouble*(request.getParameter("solarPerformance"));

Part csv\_filePart = request.getPart("solarCSV");

InputStream csv\_fileContent = csv\_filePart.getInputStream();

// Execute the Java code with the provided inputs

SolarElectricityCalculatorAndGraph calculator = new SolarElectricityCalculatorAndGraph(solarTotal, solarArea, solarEfficiency, solarPerformance, csv\_fileContent);

// Prepare response data

Map<String, Object> responseData = new HashMap<>();

responseData.put("monthsList", calculator.monthsList);

responseData.put("electricityGeneratedList", calculator.electricityGeneratedList);

// Send response as JSON

response.setContentType("application/json");

response.setCharacterEncoding("UTF-8");

response.getWriter().write(new Gson().toJson(responseData));

}

catch (Exception e) {

e.printStackTrace();

}

}

}

This above code read the user inputs (entered in the webpage) from the request sent to the server from browser, and convert it’s value to Double as user input data comes with request are always in String type and then write the value to the variables. And also read the uploaded csv file and converted it into appropriate file format. Then pass these values to the class SolarElectricityCalculatorAndGraph() (created in SolarElectricityCalculatorAndGraph.java) to perform the calculation. Then add the calculated results to the response and send the response to the client in JSON format.

**III. WindElectricityCalculatorAndGraph.java**

package com.energycalculator;

import javax.swing.\*;

import java.awt.\*;

import java.awt.geom.\*;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.util.ArrayList;

import java.util.List;

public class WindElectricityCalculatorAndGraph extends JPanel {

public List<Double> electricityGeneratedList = new ArrayList<>();

public List<String> monthsList = new ArrayList<>();

public double maxVal;

int windTotal;

double rho; // Density of the air in kg/m^3

double R; // Length of the blades (in meters)

InputStream csvFile; // Path to the CSV file containing solar hours data

public WindElectricityCalculatorAndGraph(int total\_turbines, double airDensity, double bladeLength, InputStream csv\_File) {

this.windTotal = total\_turbines;

this.rho = airDensity;

this.R = bladeLength;

this.csvFile = csv\_File;

readDataFromFile(csvFile);

}

private void readDataFromFile(InputStream csv\_file) {

String line;

String cvsSplitBy = ","; // CSV delimiter

try (BufferedReader br = new BufferedReader(new InputStreamReader(csv\_file))) {

// Skip the first line (header) as it contains column names

br.readLine();

// Read each line of the CSV file and extract the Wind Electricity data

while ((line = br.readLine()) != null) {

String[] data = line.split(cvsSplitBy);

String month = data[0]; // Assuming month is in column 1

double speed = Double.*parseDouble*(data[1]); // Assuming wind speeds are in column 2

double electricityGenerated = (0.5 \* windTotal \* rho \* 3.14159 \* Math.*pow*(R, 2) \* Math.*pow*(speed, 3)) / 1000;

// Add data to the lists

monthsList.add(month);

electricityGeneratedList.add(electricityGenerated);

}

} catch (IOException e) {

e.printStackTrace();

}

// Find max value for scaling

maxVal = electricityGeneratedList.stream().mapToDouble(Double::doubleValue).max().orElse(0);

}

public static void main(String[] args) {

}

}

**IV. SolarElectricityServlet.java**

package com.energycalculator;

import javax.servlet.\*;

import javax.servlet.annotation.MultipartConfig;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.\*;

import java.io.\*;

import java.util.\*;

import com.google.gson.Gson;

import com.energycalculator.WindElectricityCalculatorAndGraph;

*@WebServlet*(name="WindElectricityServlet", urlPatterns={"/calculate-wind-energy/"})

*@MultipartConfig*

public class WindElectricityServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// Get user input from the HTML form

int windTotal = Integer.*parseInt*(request.getParameter("windTotal"));

double airDensity = Double.*parseDouble*(request.getParameter("airDensity"));

double bladeLength = Double.*parseDouble*(request.getParameter("bladeLength"));

Part csv\_filePart = request.getPart("windCSV");

InputStream csv\_fileContent = csv\_filePart.getInputStream();

// Execute the Java code with the provided inputs

WindElectricityCalculatorAndGraph calculator = new WindElectricityCalculatorAndGraph(windTotal, airDensity, bladeLength, csv\_fileContent);

// Prepare response data

Map<String, Object> responseData = new HashMap<>();

responseData.put("monthsList", calculator.monthsList);

responseData.put("electricityGeneratedList", calculator.electricityGeneratedList);

// Send response as JSON

response.setContentType("application/json");

response.setCharacterEncoding("UTF-8");

response.getWriter().write(new Gson().toJson(responseData));

}

public static void main(String[] args) {

}

}

This above code read the user inputs (entered in the webpage) from the request sent to the server from browser, and convert it’s value to Double as user input data comes with request are always in String type and then write the value to the variables. And also read the uploaded csv file and converted it into appropriate file format. Then pass these values to the class WindElectricityCalculatorAndGraph() (created in WindElectricityCalculatorAndGraph.java) to perform the calculation. Then add the calculated results to the response and send the response to the client in JSON format.

**8.** Now open the ***web.xml***file located in the folder *src/main/webapp/WEB-INF.* And paste the below code in the file.

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**web-app** xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"* id=*"WebApp\_ID"* version=*"4.0"*>

<**servlet**>

<**servlet-name**>SolarElectricity</**servlet-name**>

<**servlet-class**>com.energycalculator.SolarElectricityServlet</**servlet-class**>

</**servlet**>

<**servlet-mapping**>

<**servlet-name**>SolarElectricity</**servlet-name**>

<**url-pattern**>/calculate-solar-energy</**url-pattern**>

</**servlet-mapping**>

<**servlet**>

<**servlet-name**>WindElectricity</**servlet-name**>

<**servlet-class**>com.energycalculator.WindElectricityServlet</**servlet-class**>

</**servlet**>

<**servlet-mapping**>

<**servlet-name**>WindElectricity</**servlet-name**>

<**url-pattern**>/calculate-wind-energy</**url-pattern**>

</**servlet-mapping**>

</**web-app**>

In the above code, an url is assigned with each created Servlet Class. For example, to execute SolarElectricityServlet the url localhost:8080/project\_name/calculate-solar-energy need to be used.

**9.** Now download the Gson package from the link <https://repo1.maven.org/maven2/com/google/code/gson/gson/2.10.1/gson-2.10.1.jar>. And move the downloaded ***gson-2.10.1.jar*** file to the folder *project\_name/src/main/webapp/WEB-INF./lib.*

**10.** Open the folder where tomcat was extracted in Step 3. Move to the *lib* folder and copy the ***servlet-api.jar*** file. Paste it to the folder *project\_name/src/main/webapp/WEB-INF./lib.*

**11**. Move the ***assets*** folder into *project\_name/src/main/* location. On Eclipse left panel right-click on project name (i.e., EnergyCalculator) and click on refresh.

**12**. On Eclipse open index.html tab, right-click on it and select Run As 🡪 Run on Server.

**13.** Finally, open the link localhost:8080/project\_name (in our case, localhost:8080/EnergyCalculator) to show the created web app. If the Tomcat server is not started the link will not work; so, make sure the Tomcat server is started, if not started start/restart the server described in the step 4.

**Deploy the webapp in a remote-server:**

**14.** Copy all the java class files from **\EnergyCalculator\build\classes\com\energycalculator** location to the location **\EnergyCalculator\src\main\webapp\WEB-INF\classes\com\energycalculator** (if the directory is not there create the directory).

**15.** Open PowerShell, shell or bash

**16.** Install flyctl. The process of the installation is given in the link. <https://fly.io/docs/hands-on/install-flyctl/>. If you are Windows user and see the error ***pwsh not found***, update PowerShell to the latest version (See the link below):

<https://learn.microsoft.com/en-us/powershell/scripting/install/installing-powershell-on-windows?view=powershell-7.4>

**17.** Open command prompt in the directory **project\_name\src\main\webapp** i.e. **EnergyCalculator\src\main\webapp**

**18**. Type **jar -cvf EnergyCalculator.war \*** and press **Enter**

**19.** Type **fly launch** and press **Enter**

**20.** Type **y** when it asked and press **Enter**

**21.** Brower will open. Sign in with GitHub

**22.** In the App name field type your web-application name

**23.** In Region select a region nearest to your country/location.

**24.** Keep other as default

**25.** Click **Confirm Settings**

**26.** Now type **fly deploy** and press **Enter**

**27.** An URL will be shown in the command prompt. Copy the URL in browser and include /project\_name (in this case it was /EnergyCalculator) at the end of the URL**,** and open the URL in browser.

**Docker** is a platform that allows you to develop, ship, and run applications inside containers. Docker is a containerization platform that you can use to package software in containers and run them on target machines. Docker containers run on any machine or virtual machine where the Docker engine is installed.

**Docker container** is a runtime environment with all the necessary components needed to run the application code without using host machine dependencies. This container runtime runs on the engine on a server, machine, or cloud instance. The engine runs multiple containers depending on the underlying resources available.

**Docker image**, or container image, is a standalone, executable file used to create a container. This container image contains all the libraries, dependencies, and files that the container needs to run. A Docker image is shareable and portable, so you can deploy the same image in multiple locations at once—much like a software binary file.

**`flyctl`** is a command-line interface (CLI) tool provided by Fly.io, a platform for deploying and running applications globally. `flyctl` allows you to manage your applications deployed on Fly.io, including deploying new versions, scaling your application, viewing logs, and managing network settings. It integrates with Docker to build and deploy applications using containerization technology. Fly.io focuses on providing a global edge computing platform, enabling developers to deploy their applications closer to end-users for lower latency and better performance.

Deploying an application globally with flyctl and Docker involves a few key steps:

* **Containerize Your Application:** Dockerize your application by creating a **Dockerfile**. This file specifies how your application should be packaged into a container image. And contains all the necessary linux commands to be executed to run the web-app in the server.
* **Configure Your flyctl Project:** Initialize a flyctl project using and configure it with your Fly.io account using **fly launch** command. This involves setting up your project name, region(s) where you want your app to be deployed.
* **Deploy Your Application:** Use the **fly deploy** command to deploy your application to the Fly.io platform. flyctl will create a Docker image on its own by using the Dockerfile located in the root directory of the project and run the container image in an isolated docker container on the server.