MACHINE LEARNING BASED AUTOMATED CONSTRUCTION PLANNING SYSTEM FOR SRI LANKA

Jathursika Linganathan - IT21223808

Final Report

Bachelor of Science (Hons) Degree in Information Technology, Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

April 2025

MACHINE LEARNING BASED AUTOMATED CONSTRUCTION PLANNING SYSTEM FOR SRI LANKA

Jathursika Linganathan - IT21223808

Final Report

Bachelor of Science (Hons) Degree in Information Technology, Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

April 2025

DECLARATION

I declare that this is my own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously publish or written by another person expect where the acknowledgement is made in the text.

Name	Student ID	Signature
Linganathan. J	IT21223808	L. Jathursika

The supervisor/s should certify the proposal report with the following declaration.
The above candidates are carrying out research for the undergraduate Dissertation under my supervision.

………………………….   …………………
Signature of the Supervisor   Date
(Mr. N.H.P. Ravi Supunya Swarnakantha)

……………………………….   …………………
Signature of the Co-Supervisor   Date
(Dr. Dharshana Kasthurirathna)

ABSTRACT

This project describes the creation of an intelligent system for cost prediction and solar panel suggestion that uses real-time user input and machine learning models to streamline and improve the solar planning process. Based on the amount of land available and the needed energy production, the system's main goal is to help users choose the best solar panel design while precisely estimating installation costs.

A Python-based Flask API that contains trained machine learning models, a Node.js backend for server-side processing, and a React.js frontend for user interaction is all integrated into the system architecture. By choosing a location and indicating the amount of land that is available, users start the procedure. The backend receives this data and uses substation data to communicate with the Flask API to determine the permitted capacity for the designated region. Users are asked to enter their desired solar capacity after the permitted capacity has been returned. The number of panels needed, their average efficiency, cost per panel, the number of panels that can fit on the designated land area, and the final anticipated installation cost are all calculated using this data.

A random forest model for panel count estimation, a gradient boosting model for efficiency estimation, and a linear regression model for cost prediction are among the machine learning models that were trained using real-world datasets. During testing, every model showed excellent accuracy, and evaluation metrics including RMSE and R2 scores validated the system's dependability. [1] [2]

Users can review their previous estimations because all recommendations made by user interaction are saved to a MongoDB database. For people and businesses thinking about installing solar panels, the results are presented in an easy-to-use interface that facilitates data-driven decision-making.

This project explains how a multi-tiered architecture that combines machine learning intelligence, backend orchestration, and user-friendly design can provide a scalable and effective solution for solar energy planning. By making complicated computations simpler, it not only promotes the use of green energy but also offers users who are concerned about their energy consumption a useful and reasonably priced tool. The system's accuracy and adaptability could be further improved in the future by integrating weather patterns, historical energy usage, and dynamic pricing models.

Keyword: - Machine Learning, Solar Panel Recommendation, Panel Cost Estimation ,Solar energy , Linear Regression

ACKNOWLEDGEMENT

First and foremost, I would like to thank **Mr. N.H.P. Ravi Supunya Swarnakantha**, my supervisor, for his constant backing, encouragement, and direction during the course of this project. His insightful observations, helpful criticism, and unwavering encouragement were crucial in determining the course and successful completion of this work.

My profound gratitude also goes out to my co-supervisor, **Dr. Dharshana Kasthurirathna**, whose technical know-how and insightful critique greatly aided in the improvement of the machine learning models and system architecture employed in this research. Under his guidance, I was able to overcome a number of obstacles and raise the standard of the research as a whole.

We are especially grateful to the Ceylon Electricity Board (CEB) for supplying the statistics on solar panels and substations that served as the system's foundation. Accurate model training and useful solution implementation were made possible by the availability of real-world data.

Finally, I would like to thank all the people and organizations that helped with this research in different ways. Your support and encouragement have been much appreciated and has been crucial to the accomplishment of this project.

TABLE OF CONTENTS

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Full Form
AI	Artificial Intelligence
API	Application Programming Interface
CEB	Ceylon Electricity Board
CSV	Comma Separated Values
DB	Database
GB	Gradient Boosting
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
kW	Kilowatt
kWh	Kilowatt-hour
ML	Machine Learning
MM	Millimeters
MSE	Mean Squared Error
MW	Megawatt
NLP	Natural Language Processing
Node.js	JavaScript Runtime built on Chrome's V8 JavaScript engine
PM	Project Manager
Python	High-level Programming Language
R^2	Coefficient of Determination (R-Squared)
RF	Random Forest
RMSE	Root Mean Squared Error
SDGs	Sustainable Development Goals
SVM	Support Vector Machine
SVR	Support Vector Regressor
UI	User Interface
UX	User Experience
Vite	Next Generation Frontend Build Tool
VS Code	Visual Studio Code
W	Watt
Wp	Watt Peak (Power rating of a solar panel under ideal conditions)
STC	Standard Test Conditions (used in solar panel testing)

Table 1 List of Abbreviation

1. INTRODUCTION

1.1 Background Study and Literature Review

1.1.1 Background Study

As the need for sustainable energy grows worldwide, solar energy has become a vital substitute for fossil fuels. Installing efficient and reasonably priced solar panels has become increasingly necessary as countries work to lessen their carbon footprint and switch to better energy sources. With year-round sunshine in places like Sri Lanka, solar energy provides a viable and sustainable answer to the nation's expanding energy requirements.

However, solar energy system deployment necessitates careful planning. To guarantee peak performance and return on investment, important factors such land size, location-based solar potential, panel efficiency, and system cost must be taken into account. These assessments are often carried out by hand or using static rule-of-thumb estimates, which frequently overlook dynamic elements like pricing, local infrastructure constraints, and real-time panel availability.

Design and implementation of renewable energy systems have been completely transformed by recent developments in **data-driven modelling** and **machine learning.** Intelligent systems have been built by researchers to use historical and current environmental data to anticipate solar irradiance, estimate panel output, and predict system efficiency. To increase forecasting and system design accuracy for solar energy, methods like ensemble learning algorithms, decision trees, and regression modelling have been used. [1] [3]

The integration of **recommendation systems with cost estimation** for solar installations is still highly lacking, notwithstanding recent developments. The majority of current solutions only address panel selection or performance prediction; they do not provide an end-to-end platform that facilitates technical and financial decision-making.

This project intends to close that gap by creating a **cost prediction and recommendation system for solar panels based on machine learning**. Using user inputs such land size and location, the system uses trained algorithms to forecast panel count, efficiency, and total cost. To process user requests and store the results in a MongoDB database, it makes use of a Flask-based Python backend that is connected with a Node.js service. React was used to build the frontend, which offers an intuitive interface for data entry, result visualization, and access to previous recommendations. [2] [3] [4] [5] [6]

The system was trained using actual information supplied by the Ceylon Electricity Board, such as substation data and a thorough inventory of solar panels, to guarantee the precision and applicability of forecasts. Several regression models, including Random Forest, Gradient Boosting, and Linear Regression, were trained using these pre-processed datasets. [7]

This solution streamlines the intricate process of solar adoption decision-making by automating the estimation process and giving consumers recommendations supported by facts. It fosters the adoption of renewable energy by making solar technology more accessible and comprehensible for users, improves informed decision-making, and lessens reliance on manual consultation.

1.1.2 Literature Review

The development of advanced technologies for solar power planning and optimization has accelerated due to the increased focus on sustainability and renewable energy. The application of data analytics and machine learning to improve the precision and dependability of solar energy forecasts has been the subject of numerous studies. This review of the literature examines the state of the art in the fields of machine learning-based cost estimating models, predictive analytics for solar system design, and solar panel performance modelling. [6]

A number of conventional methods for recommending solar panels have mostly depended on preset heuristics and deterministic models. These approaches frequently estimate the capacity or number of panels using static formulas or look-up tables based on the average solar irradiation in a given area. Despite being comparatively straightforward, these approaches are not flexible or scalable, particularly in dynamic settings where panel availability, land area, and environmental factors vary greatly. To assess PV performance, for example, Huld et al. [1] provided a solar radiation database for Europe and Africa; nevertheless, their methodology was primarily static and geographical rather than predictive.

Data-driven methods have been introduced in more recent research. Researchers like Khyati et al. [5] have shown how to model solar panel efficiency depending on variables like power rating, panel size, and climate using regression techniques like Random Forest and Support Vector Regression. By identifying trends in past data, these strategies have been demonstrated to perform better than conventional approaches, increasing the precision of photovoltaic performance predictions. [9]

Additionally, ensemble techniques like as Random Forest Regressors and Gradient Boosting have shown great efficacy in situations when the connections between input variables are complicated and nonlinear. Das et al.'s studies [3] demonstrate how these algorithms can be applied to produce economical and efficient solar power system designs that take into consideration both technical and financial factors. [2]

Although there has been much research on predicting energy output, solar recommendation systems that incorporate financial indicators are still in their infancy. The choice of panel and cost estimation are not directly related in many of the methods that are currently in use. By employing machine learning to assess both performance and cost simultaneously, research by Katiraei et al. [7] tried to close this gap. However, these models sometimes demand a significant amount of real-time commercial data, which can be challenging to collect.

An additional constraint in the extant research is the absence of user-interactive systems that integrate real-time cost estimation and intelligent advice. Many systems don't provide end users with actionable results; instead, they either anticipate sun irradiance or optimize hardware design. There is a significant disconnect between research models and practical implementation in practice.

In order to address these issues, this study offers a comprehensive platform driven by machine learning that gives customers an end-to-end solution for cost estimation and solar advice. The technology generates accurate, location-specific predictions by using real-world datasets from the Ceylon Electricity Board, including as substation-level capacity data and solar panel specs. It incorporates Gradient Boosting for efficiency forecasts, Random Forest Regressor for panel count estimation, and Linear Regression for pricing models. When these algorithms are used together, reliable performance is guaranteed even when user input varies. [11] [2]

Additionally, the system offers a comprehensive flow from user interaction to data storage and backend processing, enabling consumers to obtain personalized suggestions along with a breakdown of costs. Users can experiment with various input configurations and determine the viability and economics of their solar power systems thanks to this dynamic feedback loop.

In conclusion, even though solar energy forecasting and optimization have seen a lot of research, systems that connect technical prediction with financial planning are obviously needed. By combining real-time user interaction, precise predictions, and useful outputs, the current study expands on the literature's findings that machine learning holds great promise in this field and advances the development of intelligent, approachable, and scalable solar energy solutions.

1.2 Research Gap

Even though solar energy modeling is developing quickly and machine learning is being used more and more in photovoltaic system design, there are still a lot of unanswered questions in both current research and real-world implementations. The integration of cost estimation and real-time adaptation is frequently overlooked by the majority of current solar recommendation systems, which place a strong emphasis on either hardware optimization or energy production prediction. End users, who must also take economic viability into account while making decisions, find these systems less useful due to their compartmentalized approach. [9]

Conventional approaches usually depend on predefined rules and static datasets that are not well suited to dynamic input variables like changing land sizes, substation capabilities, or user-specific energy needs. These rule-based systems frequently fall short in terms of accuracy and adaptability, particularly when used in a variety of operational and geographic contexts. Moreover, although forecasting of solar irradiance and efficiency has been well investigated, economic factors are usually not taken into account. Users and stakeholders find it challenging to assess the practicality of solar installations in various settings due to this distance. [3]

Furthermore, even though certain machine learning models have been developed for power forecasting or panel performance, they frequently lack an interactive and thorough framework that keeps the user informed. End-to-end platforms that let users enter their precise needs (such as location, land size, and desired capacity) and get smart, real-time recommendations along with a cost breakdown are conspicuously lacking. Furthermore, the majority of research ignores infrastructure limitations that are crucial for real-world design and implementation, such as the permitted capacity limits from substations.

There are also insufficient integrated systems that incorporate frontend visualization (like dynamic dashboards or forms), middleware communication (like APIs), and backend data processing (like machine learning models). For decision-makers, companies, and the general public, these systems are less accessible due to this missing link.

The goal of this project is to close these gaps by creating a comprehensive system for cost prediction and solar suggestion that uses machine learning, real-time user input, and infrastructure limitations to deliver precise and customized solar panel options. This system differs from previous efforts in the sector due to its seamless frontend-backend interface, capacity limitations, and cost modeling.

1.3 Research Problem

The need for energy independence, growing concerns about climate change, and the depletion of fossil resources are driving a sharp increase in the demand for renewable energy worldwide. Of all the renewable energy sources, solar energy has become one of the most accessible and promising technologies. However, choosing and installing solar panels is not a simple task. Geographical location, land area available, energy needs, panel efficiency, local infrastructure capability, and cost are some of the many elements involved. It can be difficult for both individuals and organizations to make well-informed judgments based on these criteria, especially when dealing with complicated technological data and limited access to professional advice.

There aren't many smart, user-focused solutions that offer precise solar panel suggestions in the current technological environment that also take infrastructural limitations and cost-effectiveness into account. A large number of current tools are static estimators or generic calculators that don't offer recommendations that are specific to the user's location and land capacity. In addition, these tools typically lack machine learning-powered prediction capabilities and frequently fail to include real-time data, such as updated solar panel specifications or substation capacity.

Lack of a platform that facilitates a smooth transition from inquiry to recommendation and cost analysis is one of the biggest obstacles from the standpoint of the user. Users frequently have to decipher technical specifications, manually review a variety of datasets, and estimate costs either independently or with the assistance of outside consultants. Ineffective decision-making, delays, and inefficiencies result from this disjointed process. Users could overestimate the number of viable panel installations and possibly cause regulatory problems if they are not aware that the closest substation has a limited energy capacity.

In terms of technical complexity, the challenge is to integrate various data sources and predictive models into a coherent system. The performance of solar panels is affected by a variety of elements, such as power ranges, efficiency ratings, and physical dimensions, and panel specifications vary greatly. Installation costs are further influenced by land use, anticipated energy production, and market prices. Using machine learning models, scalable APIs for real-time computation and communication, historical and synthetic datasets, and a data-driven approach are all necessary to effectively handle these problems.

The gap between users and infrastructure suppliers is another major issue. For instance, the local power grid or substation, which is rarely visible to end users, determines the permissible capacity at a given place. Solar schemes that disregard these infrastructure limitations may not be feasible or compliant. Therefore, in order to provide feasible and realistic results, any useful recommendation system must incorporate these capacity limitations into its decision-making procedure.

Additionally, few machine learning algorithms have been used in real-world applications for direct user involvement, despite the fact that numerous methods have been published in academic literature for solar power forecasting or energy output prediction. Technical forecasting is the main focus of most studies, and the findings are rarely translated into useful information for regular consumers. AI in renewable energy has yet to reach its full potential due to a lack of end-to-end platforms that connect data science with user experience. [3]

Cost estimation is another area where the disparity exists. Adoption of solar energy is significantly hampered by cost, particularly for small and medium-sized businesses or in developing nations. The financial feasibility of solar investments cannot be ascertained by users without a precise and customized cost prediction. The majority of current methods provide rough estimates without taking substation limitations, efficiency, land utilization, or panel variance into consideration. Users either overspend on superfluous capacity or underuse the infrastructure and space that are available as a result.

Therefore, the central research problem addressed by this project is: **How can we develop an intelligent, integrated solar panel recommendation and cost estimation system that uses machine learning to generate real-time, location-based, and infrastructure-aware recommendations for end users?**

To solve this problem, this research proposes a novel system that:

- Accepts user inputs such as location and land size.
- Determines the allowable capacity based on substation constraints.
- Applies trained machine learning models to estimate panel efficiency, panel count, and cost.
- Enables users to enter desired capacity and receive cost estimations.
- Stores and retrieves recommendations through a structured database.
- Uses a Node.js-based backend for API routing and Flask-based microservice for machine learning processing.
- Displays results via a user-friendly React frontend.

In conclusion, this research challenge captures the need for an intelligent, easily accessible system that links people to solar solutions in a precise, economical, and customized way. Solving this issue will support the larger objectives of sustainability, digital transformation, and renewable energy optimization in addition to facilitating well-informed decision-making for solar adoption.

1.4 Research Objectives

1.1.1 Main objectives

The primary goal of this project is to build and create a clever system for cost estimation and solar panel suggestion based on machine learning. By taking into account factors including land size, substation capacity, panel efficiency, and cost, the system seeks to give customers precise, location-based recommendations for solar installations, enabling them to make well-informed decisions for the adoption of solar energy that is both economical and sustainable.

1.1.2 Specific objective

To gather and preprocess pertinent datasets on substation capacity and solar panel parameters.

- To create machine learning models for cost estimation, necessary panel count, and solar panel efficiency prediction.
- To incorporate user input for real-time personalized output, such as location and land size, into the recommendation system.
- To use React to create a web-based frontend with Node.js and Flask for backend services.
- To make it possible for suggestion records to be stored and retrieved for analysis and traceability in a MongoDB database.
- To guarantee system usability and assess machine learning model performance using suitable metrics.

1.1.3 Business objective

a) Encourage Cost-Effective Renewable Adoption:
One of the main objectives of the company is to make the transition to solar energy less complicated and uncertain financially. The system enables users to make cost-effective

energy decisions by providing accurate cost estimates and optimal panel configurations, which encourages small-to-medium businesses and people to embrace solar technology more widely.

b) Enhance Decision-Making Through Data Intelligence:

   By providing actionsable recommendations supported by data and machine learning insights, the platform helps users and infrastructure limitations (such as substation capabilities) communicate more effectively. This lowers planning errors, guarantees adherence to energy infrastructure constraints, and improves operational transparency.

c) Offer the Energy Market Scalable and User-Friendly Tools:

   Offering a scalable software-as-a-service (SaaS) solution to renewable energy stakeholders is the long-term goal. Solar providers, utilities, and consultants can use the system to expedite the client onboarding process, improving business efficiency and service quality, thanks to its modular backend and user-friendly interface.

2. METHODOLOGY

2.1 Methodology

The suggested strategy for recommending solar panels based on available land capacity and substation limits was created using a methodical and useful approach. The development process used a hybrid methodology that included frontend interface creation using React.js, Flask-based API integration, machine learning model building, and data preprocessing. This section lists the main stages of the implementation process.

2.1.1 Feasibility study

The project began with a feasibility study that examined the system's technological, operational, and financial components. The Ceylon Electricity Board assisted in verifying key data sources, including substation capacity statistics and databases of synthetic solar panels. To make sure the suggested solution would be useful in practice, a range of stakeholders were contacted, including engineers and domain specialists. The initial planning process was divided into milestone phases, which included frontend integration, model training, API development, and dataset preparation

2.1.2 Requirement gathering

In order to determine the system's scope, functional requirements were obtained. Among these were:

- Processing capacity limitations at the substation level.
- Making recommendations for appropriate solar panel arrangements.
- Estimating the number of panels based on permitted capacity and land area.
- Cost estimation based on changes in market prices.

Including a web interface in the system.
System performance, usability, scalability, and maintainability were all considered non-functional needs. Stakeholder input and iterative improvement were used to validate the requirements.

2.1.3 Data Collection

In order to build the model and system intelligence, the study required gathering and merging two datasets:

i) The Solar Panel Specification Dataset was a synthetic dataset that replicated manufacturer standards. Dimensions (height, width), Series Power Range, Efficiency at STC, and price details were among the fields it contained. The panels that are available on the market are represented by these parameters.
ii) The Ceylon Electricity Board (CEB) supplied the substation capacity dataset, which identified the permitted solar capacities for each location. It facilitated the correlation between the user's chosen location and the actual limits of solar installations.

The Python environment was used to clean and alter the two datasets once they were imported in Excel format.

2.1.4 Data Preprocessing

- Several preprocessing steps were conducted in order to get the data ready for modelling.
- Using removal and replacement to deal with missing or inconsistent values.
- Deriving the minimum and maximum power values (W) from a single power range column.
- Panel size is calculated by multiplying height by width and translating mm^2 to m^2.
- Standardizing column renaming and formatting.

- To increase training stability, standardize numerical features using Standard Scaler. It is necessary to map and encode categorical information, including location names, in lowercase to guarantee uniformity among services.

By ensuring dataset compatibility, these preparation processes enhanced machine learning models' capacity for generalization.

2.1.5 Feature Engineering

Feature engineering was performed to define and extract meaningful predictors:

- **Input Features**:
    - Min Power (W)
    - Max Power (W)
    - Allowed Capacity (W)
    - Panel Efficiency (%) At STC
    - Price
- **Target Outputs**:
    - Number of Panels
    - Efficiency
    - Final Cost
    - Panel Size

These features were selected based on domain relevance and their influence on solar energy system design and economics.

2.1.6 Model Selection

To meet the prediction requirements for panel count, cost estimation, and efficiency, a number of supervised regression models were investigated:

1. Random Forest Regressor: Selected for its ensemble averaging resilience in predicting the number of panels.
2. Gradient Boosting Regressor: Because it can handle non-linear relationships, this regression is used to forecast efficiency.
3. Because of its explainability and strong performance with scaled data, linear regression is the method of choice for cost prediction.
4. In order to compare accuracy with ensemble models, the Decision Tree Regressor served as a baseline.
5. Support Vector Regressor (SVR): SVR's ability to generalize tiny feature sets was tested.

RMSE, R2 score, and cross-validated scores were among the performance indicators. [11]

2.1.7 Model Training and Evaluation

For testing and training, the dataset was divided in a 70:30 ratio. The models were assessed using:

- The robustness of the model was ensured via five-fold cross-validation.
- Standard Scaler: Made ensuring that models were trained using input features that were standardized.
- **Evaluation Metrics**:
    - The Random Forest Regressor demonstrated near-perfect panel count estimate with an RMSE of around 0.0000, according to the evaluation metrics.
    - Gradient Boosting demonstrated excellent predictive performance for efficiency, with an RMSE of 2.38.

- Linear Regression validated the accuracy of cost estimation with an R2 score of 0.98.

These outcomes validated the model's dependability prior to implementation.

2.1.8 System Architecture

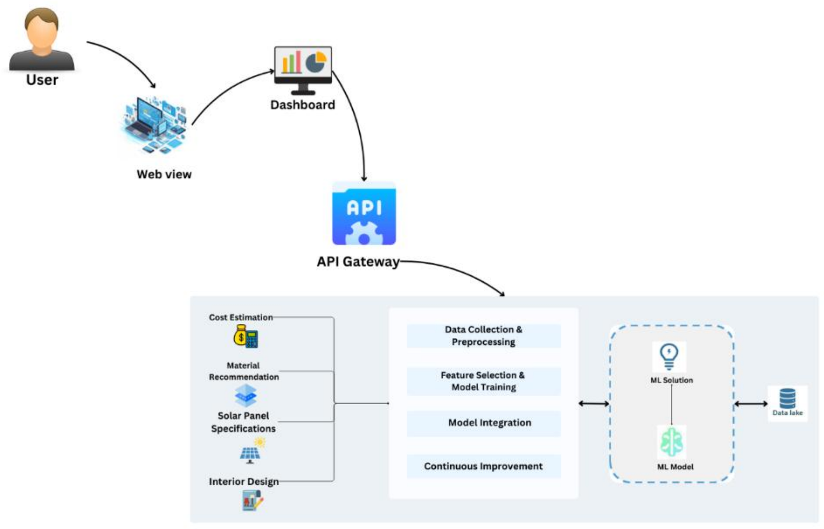


Figure 1 Overall System Architecture

The overall system comprises several interlinked modules:

- **Frontend**: Built with React (Vite), it offers a dynamic UI for user interaction.
- **Backend API**: Developed with Node.js and Express, it handles business logic and routes.
- **Machine Learning API**: Powered by Flask (Python), it hosts models and processes inference logic.
- **Database Layer**: MongoDB stores user input history, model outputs, and recommendation results.

This architecture supports scalability and modular updates.

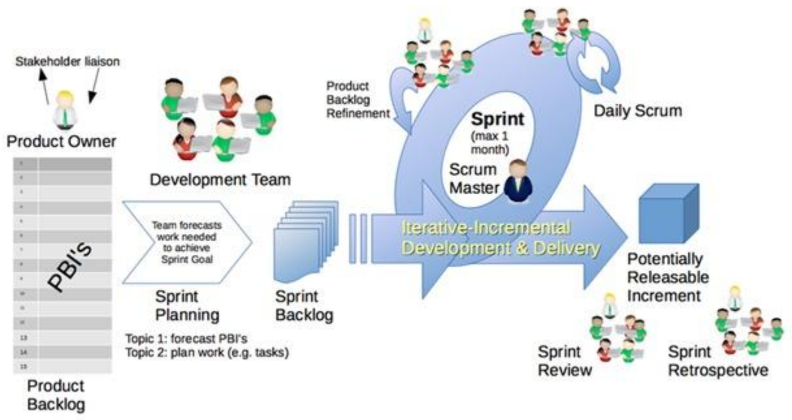


Figure 2 Followed Architecture

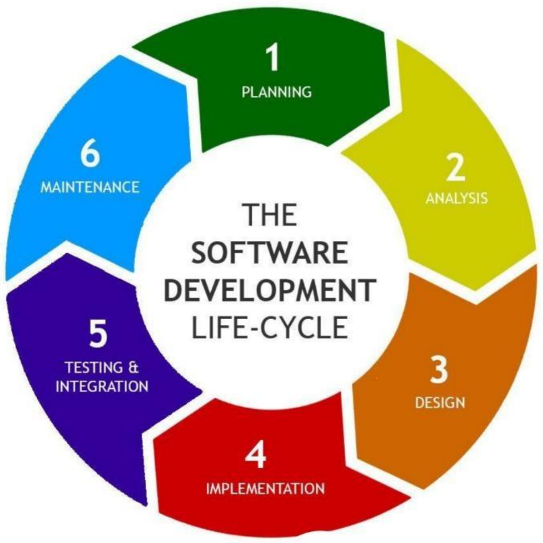


Figure 3 Followed Software Development Life cycle

2.1.9 Technology Stack

Layer	**Tools and Frameworks**
Frontend	React, Vite, Tailwind CSS
Backend	Node.js, Express.js

ML API	Python, Flask, scikit-learn, pandas
Database	MongoDB (NoSQL)
Development	Google Colab, Jupyter Notebook

Table 2 Technology stack

2.1.10 System Workflow

The end-to-end workflow comprises:

1. User input: Using the frontend, the user chooses a location and inputs the size of the land.
2. Step 1-Capacity Estimation: The Flask API receives a request from the backend and provides the location's permitted capacity.
3. Step 2: Cost Estimation: The user supplies the capacity they want. Forecasted panel count, average efficiency, and total cost are returned by Flask.
4. Data Storage: The Node.js backend stores all responses in MongoDB.
5. Results Display: For user clarity, the output is displayed as structured cards or tables by the frontend.

A very sophisticated and dynamic recommendation experience is guaranteed by this cycle.

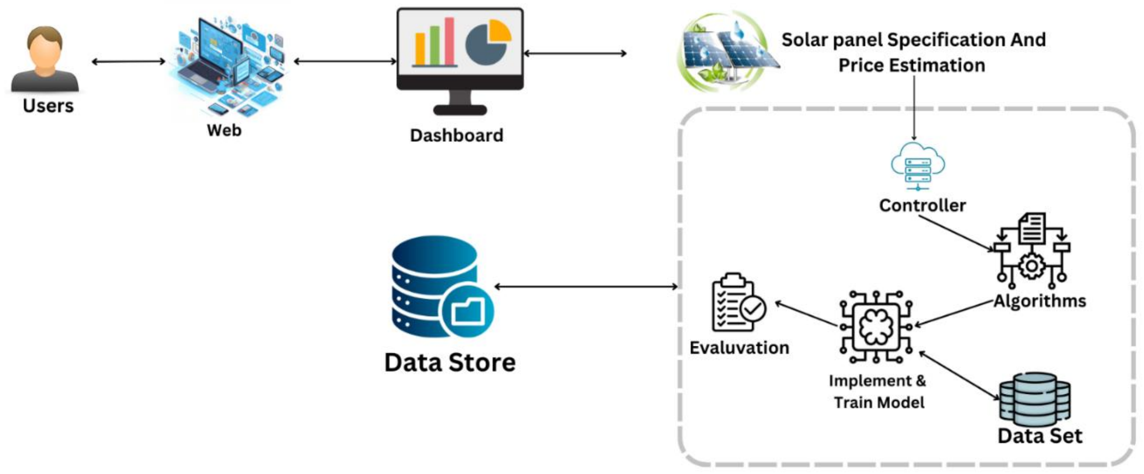
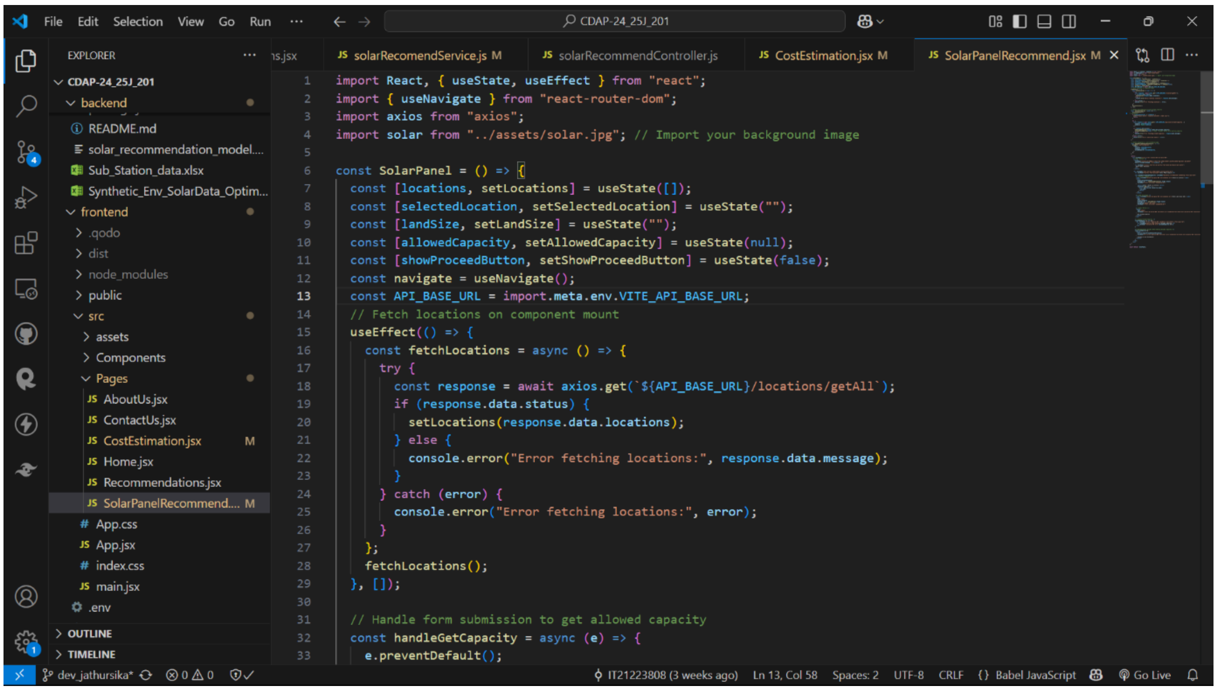
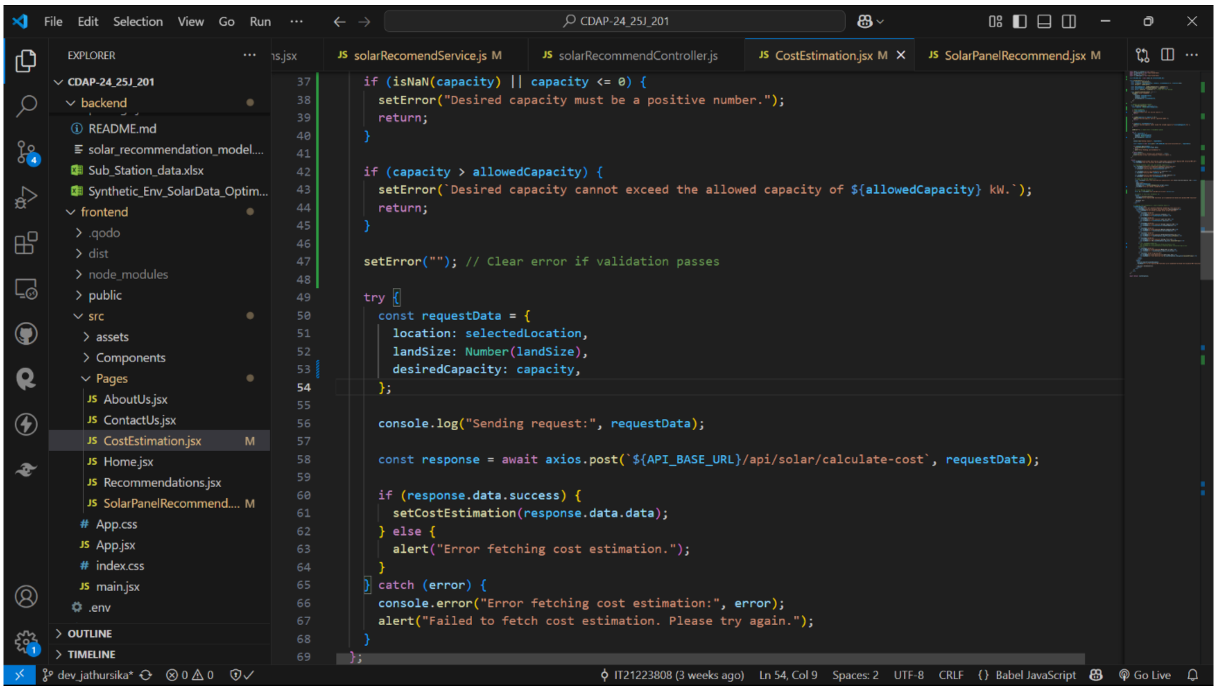


Figure 4 Component Architecture



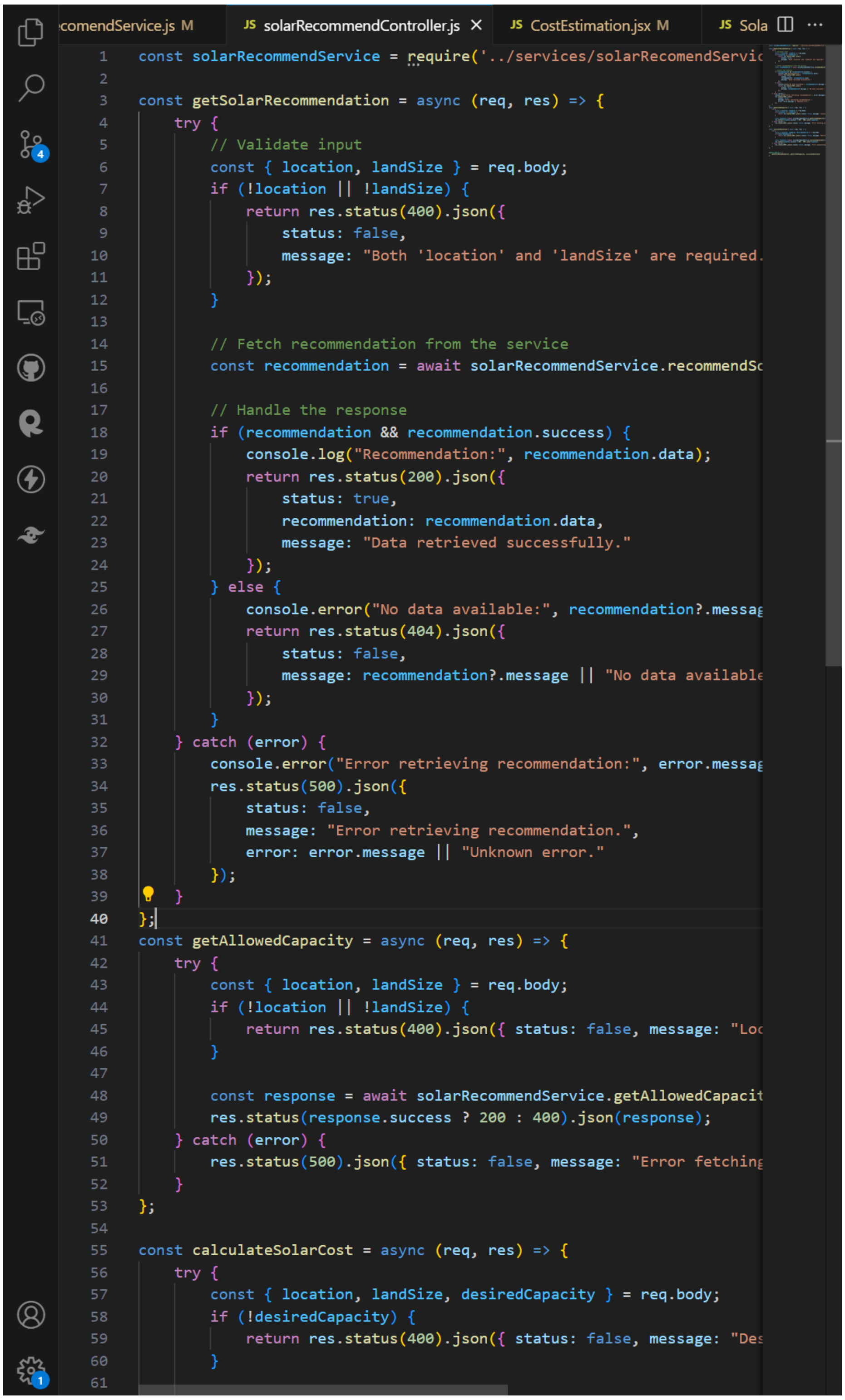
```
import React, { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import axios from "axios";
import solar from "../assets/solar.jpg"; // Import your background image

const SolarPanel = () => {
  const [locations, setLocations] = useState([]);
  const [selectedLocation, setSelectedLocation] = useState("");
  const [landSize, setLandSize] = useState("");
  const [allowedCapacity, setAllowedCapacity] = useState(null);
  const [showProceedButton, setShowProceedButton] = useState(false);
  const navigate = useNavigate();
  const API_BASE_URL = import.meta.env.VITE_API_BASE_URL;
  // Fetch locations on component mount
  useEffect(() => {
    const fetchLocations = async () => {
      try {
        const response = await axios.get(`${API_BASE_URL}/locations/getAll`);
        if (response.data.status) {
          setLocations(response.data.locations);
        } else {
          console.error("Error fetching locations:", response.data.message);
        }
      } catch (error) {
        console.error("Error fetching locations:", error);
      }
    };
    fetchLocations();
  }, []);

  // Handle form submission to get allowed capacity
  const handleGetCapacity = async (e) => {
    e.preventDefault();
```

Figure 5 Frontend code 1



```
    if (isNaN(capacity) || capacity <= 0) {
      setError("Desired capacity must be a positive number.");
      return;
    }

    if (capacity > allowedCapacity) {
      setError(`Desired capacity cannot exceed the allowed capacity of ${allowedCapacity} kW.`);
      return;
    }

    setError(""); // Clear error if validation passes

    try {
      const requestData = {
        location: selectedLocation,
        landSize: Number(landSize),
        desiredCapacity: capacity,
      };

      console.log("Sending request:", requestData);

      const response = await axios.post(`${API_BASE_URL}/api/solar/calculate-cost`, requestData);

      if (response.data.success) {
        setCostEstimation(response.data.data);
      } else {
        alert("Error fetching cost estimation.");
      }
    } catch (error) {
      console.error("Error fetching cost estimation:", error);
      alert("Failed to fetch cost estimation. Please try again.");
    }
  };
```

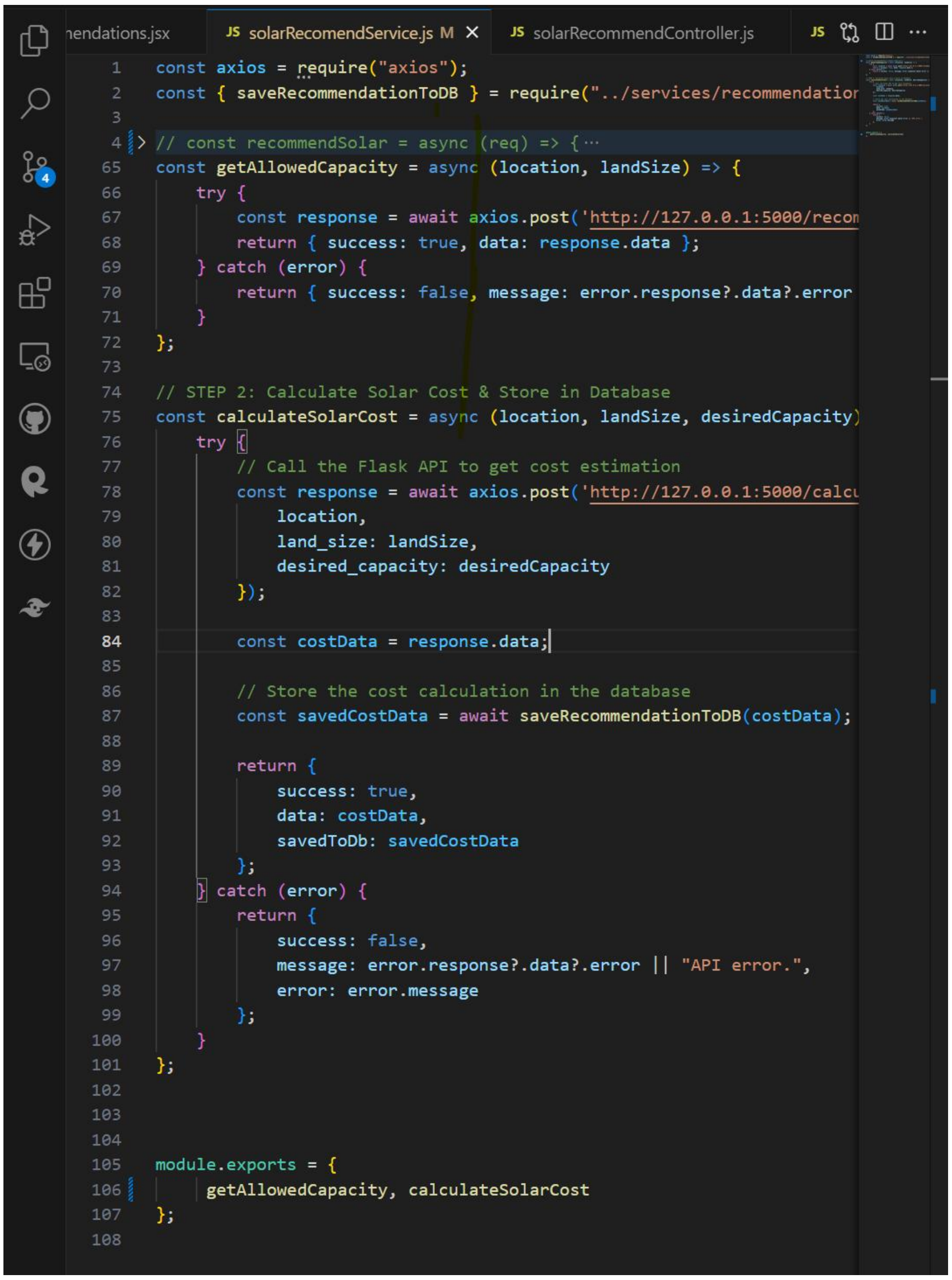
Figure 6 Frontend code 2



```
const solarRecommendService = require('../services/solarRecomendServic

const getSolarRecommendation = async (req, res) => {
    try {
        // Validate input
        const { location, landSize } = req.body;
        if (!location || !landSize) {
            return res.status(400).json({
                status: false,
                message: "Both 'location' and 'landSize' are required.
            });
        }

        // Fetch recommendation from the service
        const recommendation = await solarRecommendService.recommendSc

        // Handle the response
        if (recommendation && recommendation.success) {
            console.log("Recommendation:", recommendation.data);
            return res.status(200).json({
                status: true,
                recommendation: recommendation.data,
                message: "Data retrieved successfully."
            });
        } else {
            console.error("No data available:", recommendation?.messag
            return res.status(404).json({
                status: false,
                message: recommendation?.message || "No data available
            });
        }
    } catch (error) {
        console.error("Error retrieving recommendation:", error.messag
        res.status(500).json({
            status: false,
            message: "Error retrieving recommendation.",
            error: error.message || "Unknown error."
        });
    }
};
const getAllowedCapacity = async (req, res) => {
    try {
        const { location, landSize } = req.body;
        if (!location || !landSize) {
            return res.status(400).json({ status: false, message: "Loc
        }

        const response = await solarRecommendService.getAllowedCapacit
        res.status(response.success ? 200 : 400).json(response);
    } catch (error) {
        res.status(500).json({ status: false, message: "Error fetching
    }
};

const calculateSolarCost = async (req, res) => {
    try {
        const { location, landSize, desiredCapacity } = req.body;
        if (!desiredCapacity) {
            return res.status(400).json({ status: false, message: "Des
        }

```

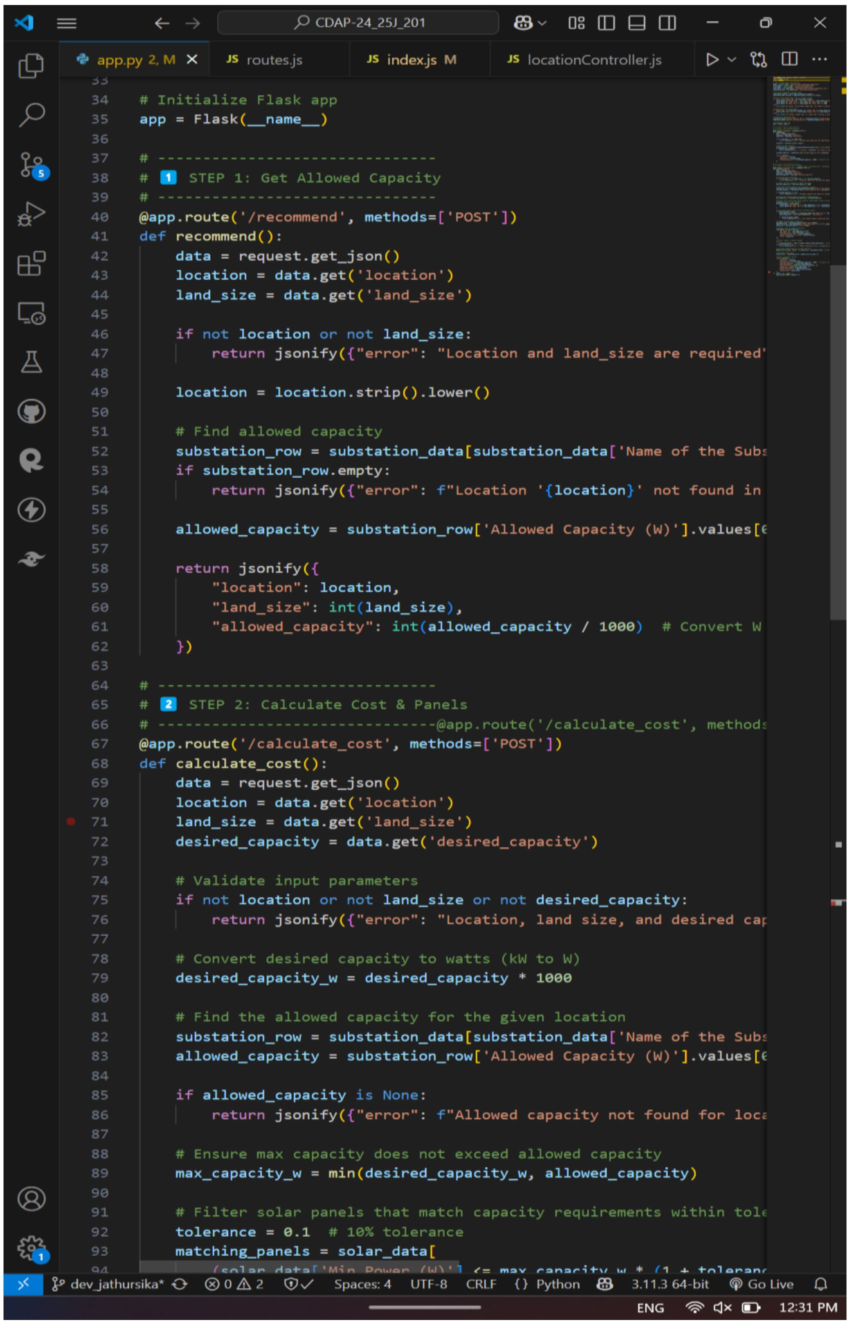
z

Figure 7



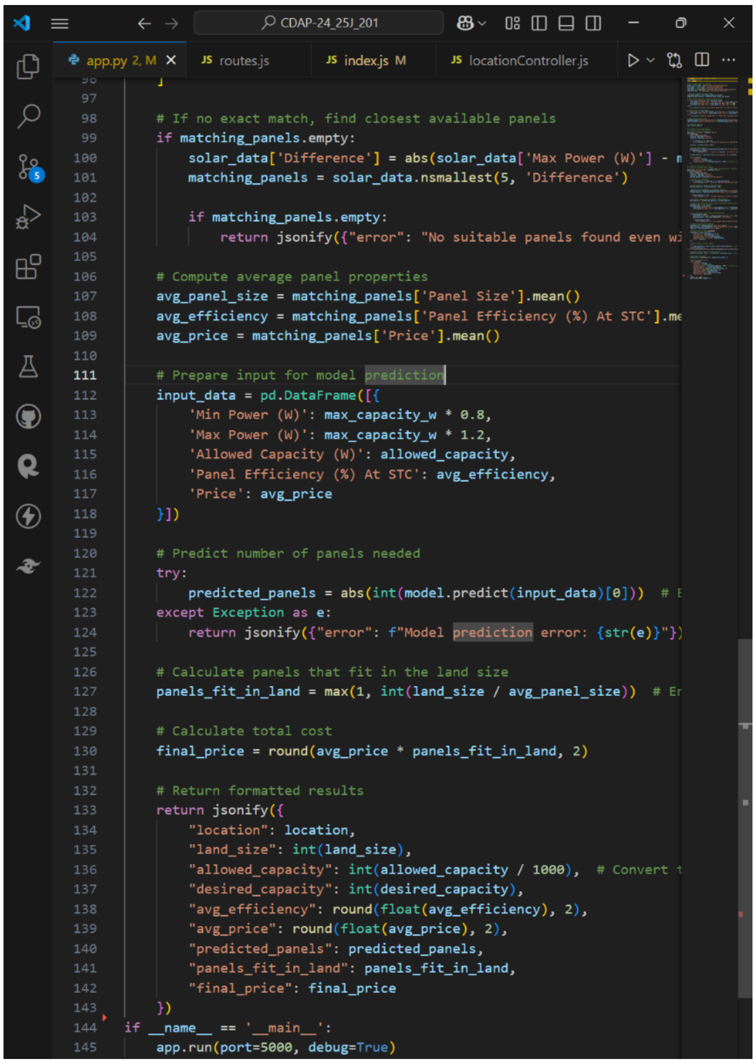
```javascript
const axios = require("axios");
const { saveRecommendationToDB } = require("../services/recommendatior

// const recommendSolar = async (req) => {···
const getAllowedCapacity = async (location, landSize) => {
    try {
        const response = await axios.post('http://127.0.0.1:5000/recom
        return { success: true, data: response.data };
    } catch (error) {
        return { success: false, message: error.response?.data?.error
    }
};

// STEP 2: Calculate Solar Cost & Store in Database
const calculateSolarCost = async (location, landSize, desiredCapacity)
    try {
        // Call the Flask API to get cost estimation
        const response = await axios.post('http://127.0.0.1:5000/calcu
            location,
            land_size: landSize,
            desired_capacity: desiredCapacity
        });

        const costData = response.data;

        // Store the cost calculation in the database
        const savedCostData = await saveRecommendationToDB(costData);

        return {
            success: true,
            data: costData,
            savedToDb: savedCostData
        };
    } catch (error) {
        return {
            success: false,
            message: error.response?.data?.error || "API error.",
            error: error.message
        };
    }
};



module.exports = {
    getAllowedCapacity, calculateSolarCost
};

```

Figure 8



```
# Initialize Flask app
app = Flask(__name__)

# ------------------------------
# 1️⃣ STEP 1: Get Allowed Capacity
# ------------------------------
@app.route('/recommend', methods=['POST'])
def recommend():
    data = request.get_json()
    location = data.get('location')
    land_size = data.get('land_size')

    if not location or not land_size:
        return jsonify({"error": "Location and land_size are required'

    location = location.strip().lower()

    # Find allowed capacity
    substation_row = substation_data[substation_data['Name of the Subs
    if substation_row.empty:
        return jsonify({"error": f"Location '{location}' not found in

    allowed_capacity = substation_row['Allowed Capacity (W)'].values[0

    return jsonify({
        "location": location,
        "land_size": int(land_size),
        "allowed_capacity": int(allowed_capacity / 1000)  # Convert W
    })

# ------------------------------
# 2️⃣ STEP 2: Calculate Cost & Panels
# ------------------------------@app.route('/calculate_cost', methods
@app.route('/calculate_cost', methods=['POST'])
def calculate_cost():
    data = request.get_json()
    location = data.get('location')
    land_size = data.get('land_size')
    desired_capacity = data.get('desired_capacity')

    # Validate input parameters
    if not location or not land_size or not desired_capacity:
        return jsonify({"error": "Location, land size, and desired cap

    # Convert desired capacity to watts (kW to W)
    desired_capacity_w = desired_capacity * 1000

    # Find the allowed capacity for the given location
    substation_row = substation_data[substation_data['Name of the Subs
    allowed_capacity = substation_row['Allowed Capacity (W)'].values[0

    if allowed_capacity is None:
        return jsonify({"error": f"Allowed capacity not found for loca

    # Ensure max capacity does not exceed allowed capacity
    max_capacity_w = min(desired_capacity_w, allowed_capacity)

    # Filter solar panels that match capacity requirements within tole
    tolerance = 0.1  # 10% tolerance
    matching_panels = solar_data[
        (solar_data['Min Power (W)'] <= max_capacity_w * (1 + toleranc
```

Figure 9 app.py capture 1



```python
    ]

    # If no exact match, find closest available panels
    if matching_panels.empty:
        solar_data['Difference'] = abs(solar_data['Max Power (W)'] - m
        matching_panels = solar_data.nsmallest(5, 'Difference')

        if matching_panels.empty:
            return jsonify({"error": "No suitable panels found even wi

    # Compute average panel properties
    avg_panel_size = matching_panels['Panel Size'].mean()
    avg_efficiency = matching_panels['Panel Efficiency (%) At STC'].me
    avg_price = matching_panels['Price'].mean()

    # Prepare input for model prediction
    input_data = pd.DataFrame([{
        'Min Power (W)': max_capacity_w * 0.8,
        'Max Power (W)': max_capacity_w * 1.2,
        'Allowed Capacity (W)': allowed_capacity,
        'Panel Efficiency (%) At STC': avg_efficiency,
        'Price': avg_price
    }])

    # Predict number of panels needed
    try:
        predicted_panels = abs(int(model.predict(input_data)[0]))  # E
    except Exception as e:
        return jsonify({"error": f"Model prediction error: {str(e)}"})

    # Calculate panels that fit in the land size
    panels_fit_in_land = max(1, int(land_size / avg_panel_size))  # Er

    # Calculate total cost
    final_price = round(avg_price * panels_fit_in_land, 2)

    # Return formatted results
    return jsonify({
        "location": location,
        "land_size": int(land_size),
        "allowed_capacity": int(allowed_capacity / 1000),  # Convert t
        "desired_capacity": int(desired_capacity),
        "avg_efficiency": round(float(avg_efficiency), 2),
        "avg_price": round(float(avg_price), 2),
        "predicted_panels": predicted_panels,
        "panels_fit_in_land": panels_fit_in_land,
        "final_price": final_price
    })
if __name__ == '__main__':
    app.run(port=5000, debug=True)
```

Figure 10 app.py capture 2

2.1.11 Validation and Testing

A functional and dependable system was guaranteed by several testing stages:

- **Unit Testing:** Separate ML operations and API routes are examined separately.
- **Integration Testing:** Confirmed data flow between ML, frontend, and backend APIs.
- **Performance testing:** Response times under simulated loads are benchmarked.
- **User Acceptance Testing (UAT):** This method assessed the results' interpretability, input flow, and usability using a sample of users.

Each test cycle was followed by performance improvement and bug remedies.

2.1.12 Limitations

Even while the technique works, it has certain limitations:

- It ignores current weather and solar irradiance.
- There is no integration of real-time stock checks, and the inventory of available panels is static.
- Custom configurations that go beyond size and placement are not supported.

These dimensions might be added in later iterations for more usefulness.

2.1.13 Ethical Considerations

Maintaining ethical integrity involved:

- Making sure that no personally identifying information was gathered.
- Making use of permitted or public datasets.
- When appropriate, referencing outside sources and algorithms.
- Steer clear of advice that are overly dramatic or overdone.

Disclaimers on accuracy variance resulting from synthetic data sources were included in the models' deployment.

3. Testing & Implementation

3.1 Functional Testing with Sample Inputs and Outputs

To ensure the core functionalities of the Solar Panel Recommendation and Cost Estimation System operate correctly, several test cases were conducted. Each case is documented below with user inputs, system flow, and observed outputs.

Test Case 1: Get Allowed Capacity Based on Location and Land Size

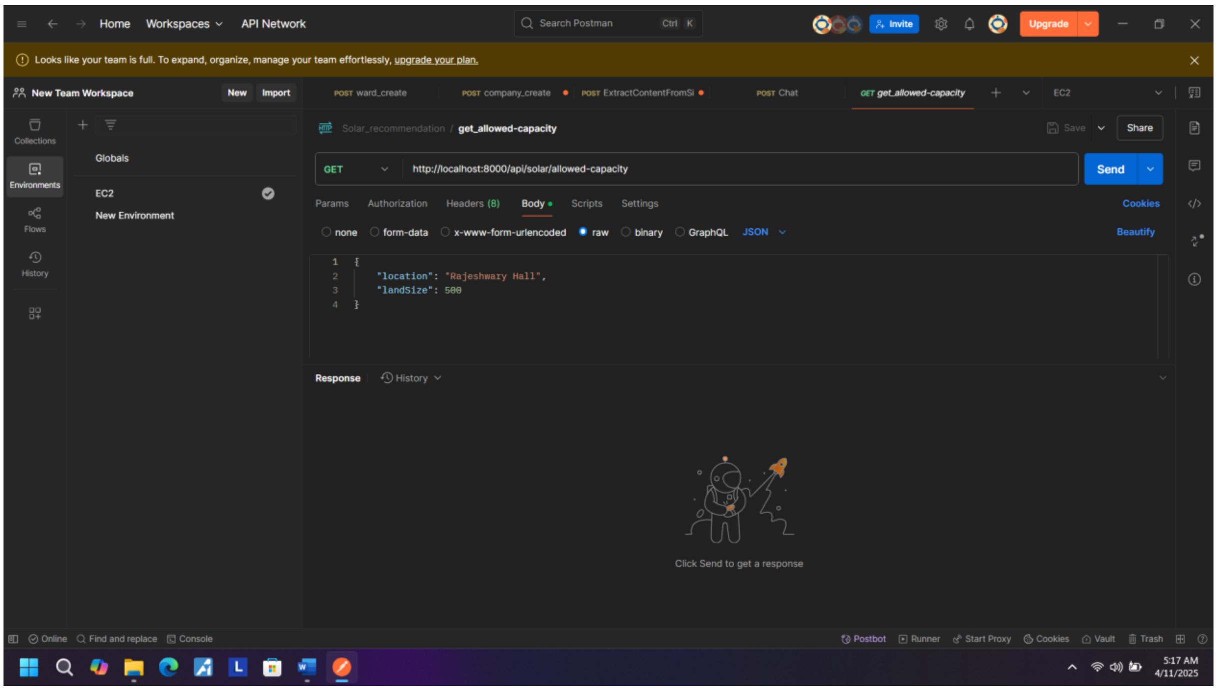
- **API Endpoint**: /api/solar/allowed-capacity
- **Input**:

{

"location": "Rajeshwary Hall",

"landSize": 500

}



Looks like your team is full. To expand, organize, manage your team effortlessly, upgrade your plan.

Solar_recommendation / get_allowed-capacity

GET http://localhost:8000/api/solar/allowed-capacity

```
{
  "location": "Rajeshwary Hall",
  "landSize": 500
}
```

Click Send to get a response

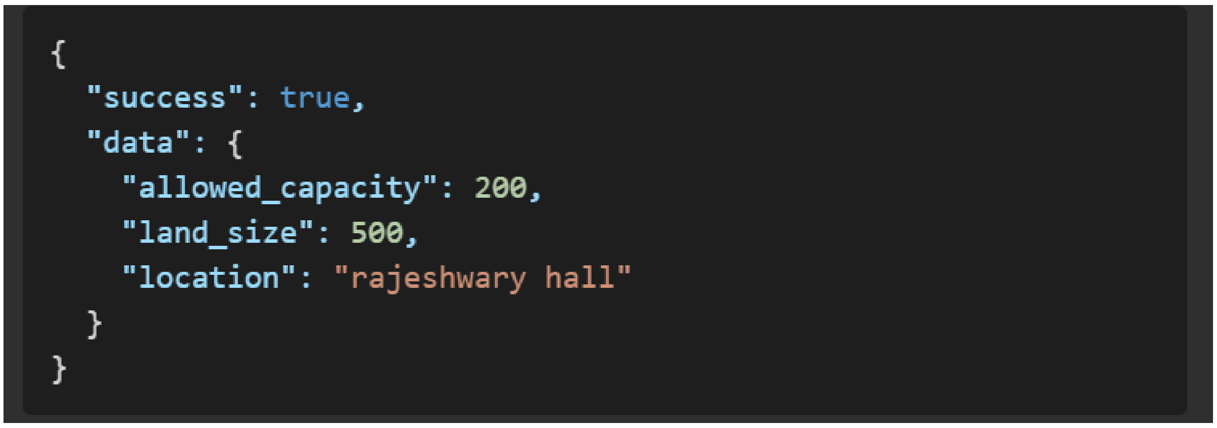
Figure 11 Get Allowed Capacity Based on Location and Land Size

- **Expected Output**:

```
{
  "success": true,
  "data": {
    "allowed_capacity": 200,
    "land_size": 500,
    "location": "rajeshwary hall"
  }
}
```

- **Actual Output**: Matches expected output.

Result: Pass



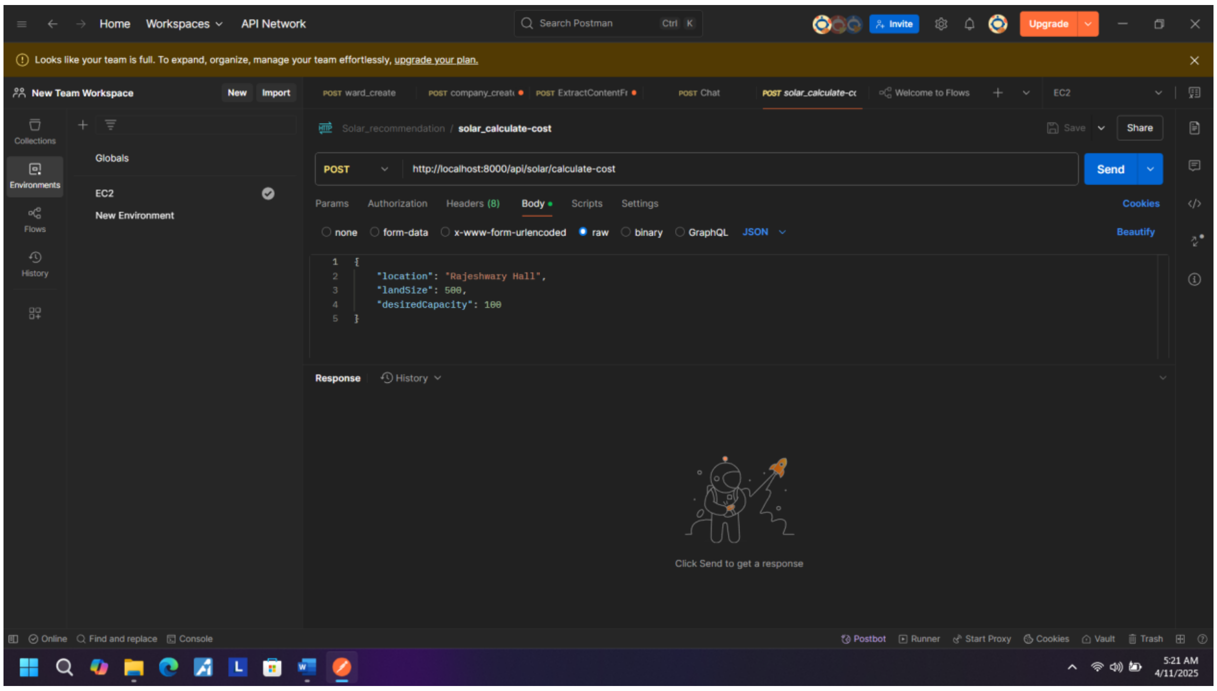
```
{
  "success": true,
  "data": {
    "allowed_capacity": 200,
    "land_size": 500,
    "location": "rajeshwary hall"
  }
}
```

Figure 12 Get Allowed Capacity Based on Location and Land Size -output

Test Case 2: Get Cost Estimation Based on Desired Capacity

- **API Endpoint**: /api/solar/calculate-cost
- **Input**:

```
{
  "location": "Rajeshwary Hall",
  "land_size": 500,
  "desired_capacity": 100
}
```



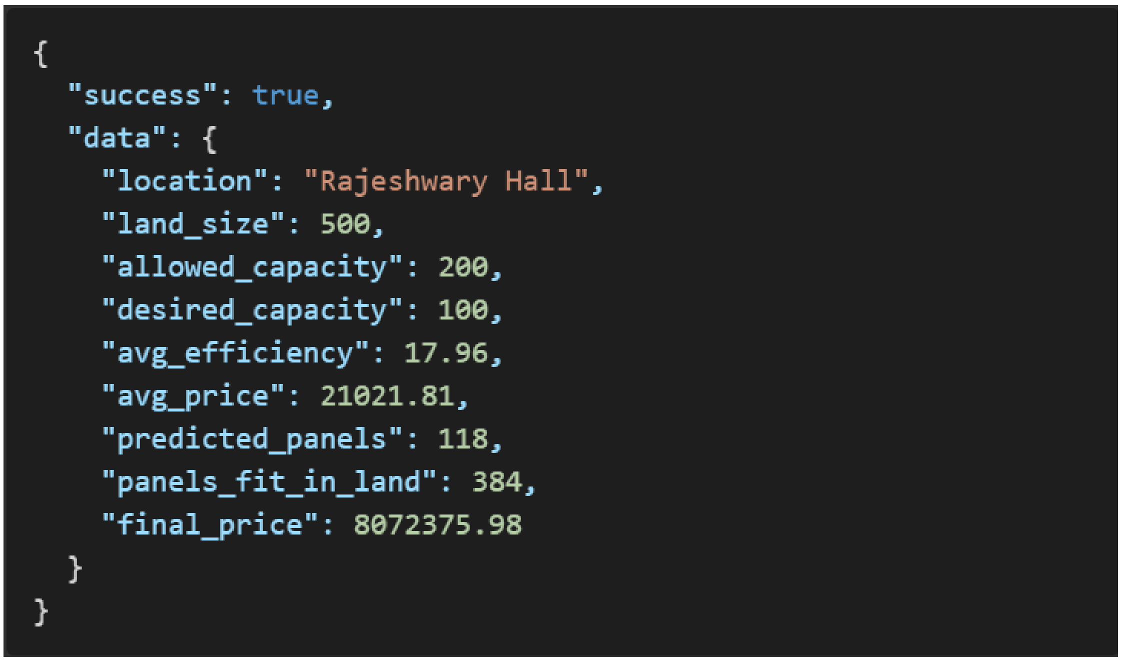
POST http://localhost:8000/api/solar/calculate-cost

```
{
  "location": "Rajeshwary Hall",
  "landSize": 500,
  "desiredCapacity": 100
}
```

Figure 13 Get Cost Estimation Based on Desired Capacity

- **Expected Output**: A structured prediction including efficiency, price, panel count, and final cost.
- **Actual Output**:

```
{
  "success": true,
  "data": {
    "location": "Rajeshwary Hall",
    "land_size": 500,
    "allowed_capacity": 200,
    "desired_capacity": 100,
    "avg_efficiency": 17.96,
    "avg_price": 21021.81,
    "predicted_panels": 118,
    "panels_fit_in_land": 384,
    "final_price": 8072375.98
  }
}
```



```
{
  "success": true,
  "data": {
    "location": "Rajeshwary Hall",
    "land_size": 500,
    "allowed_capacity": 200,
    "desired_capacity": 100,
    "avg_efficiency": 17.96,
    "avg_price": 21021.81,
    "predicted_panels": 118,
    "panels_fit_in_land": 384,
    "final_price": 8072375.98
  }
}
```

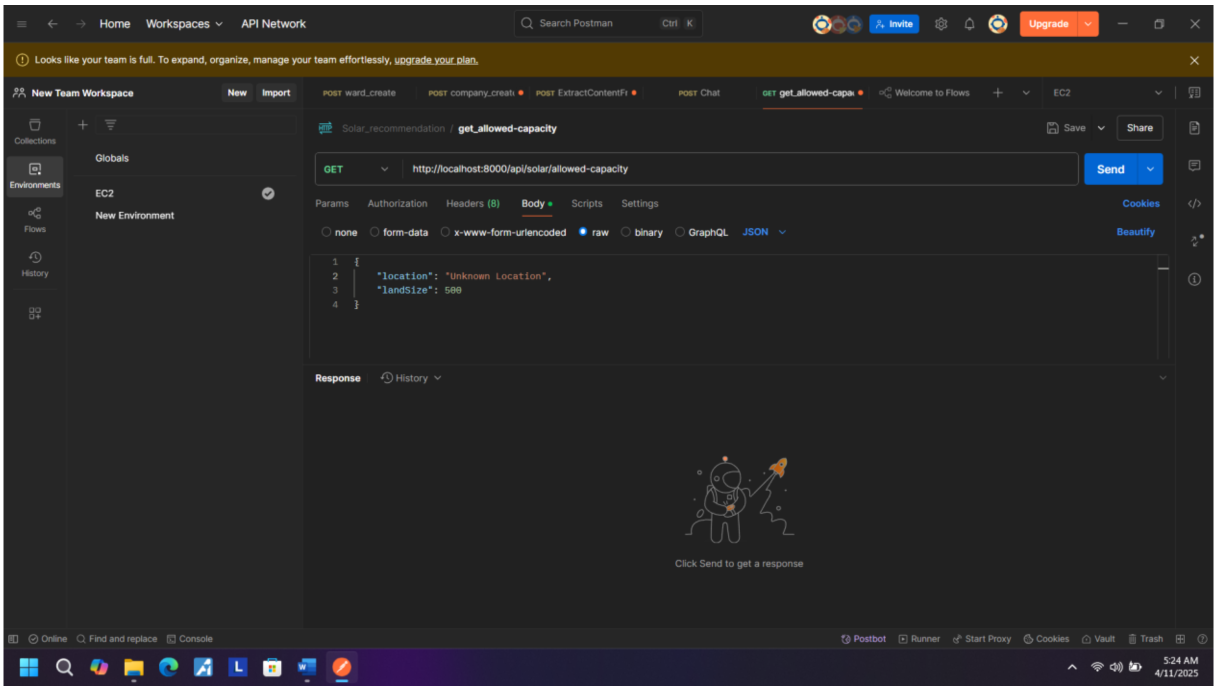
Figure 14 Get Cost Estimation Based on Desired Capacity-output

Result: Pass

Test Case 3: Invalid Location Input

- **Input**:

```
{
  "location": "Unknown Location",
  "landSize": 500
}
```


Solar_recommendation / get_allowed-capacity

GET http://localhost:8000/api/solar/allowed-capacity

```
{
    "location": "Unknown Location",
    "landSize": 500
}
```

Click Send to get a response

Figure 15 Invalid Location Input

- **Expected Output**: Error response due to location mismatch.
- **Actual Output**:

{

"error": "Location 'unknown location' not found in substation data."

}

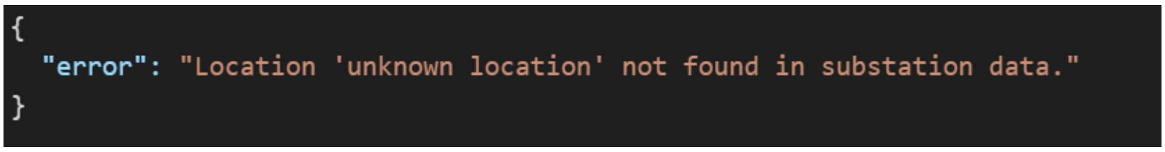
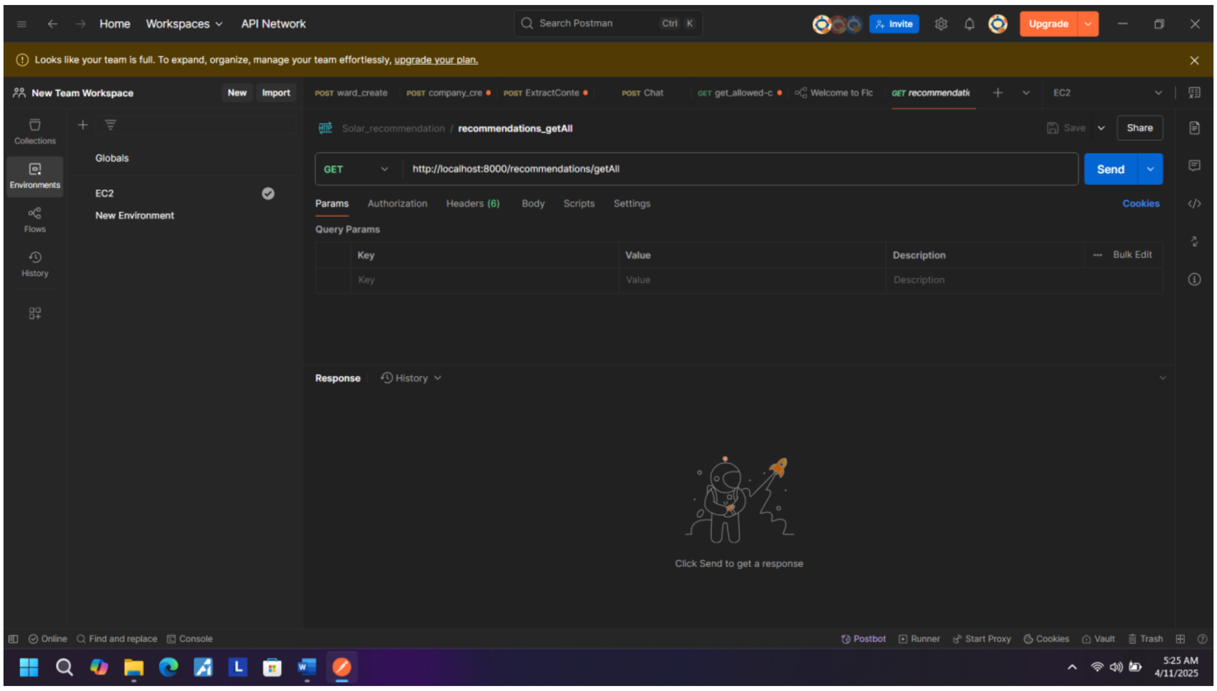

```
{
  "error": "Location 'unknown location' not found in substation data."
}
```

Figure 16 Invalid Location Input-output

Result: Pass

Test Case 4: Display Recommendations List

- **API Endpoint**: /recommendations/getAll


Solar_recommendation / recommendations_getAll

GET http://localhost:8000/recommendations/getAll

Query Params

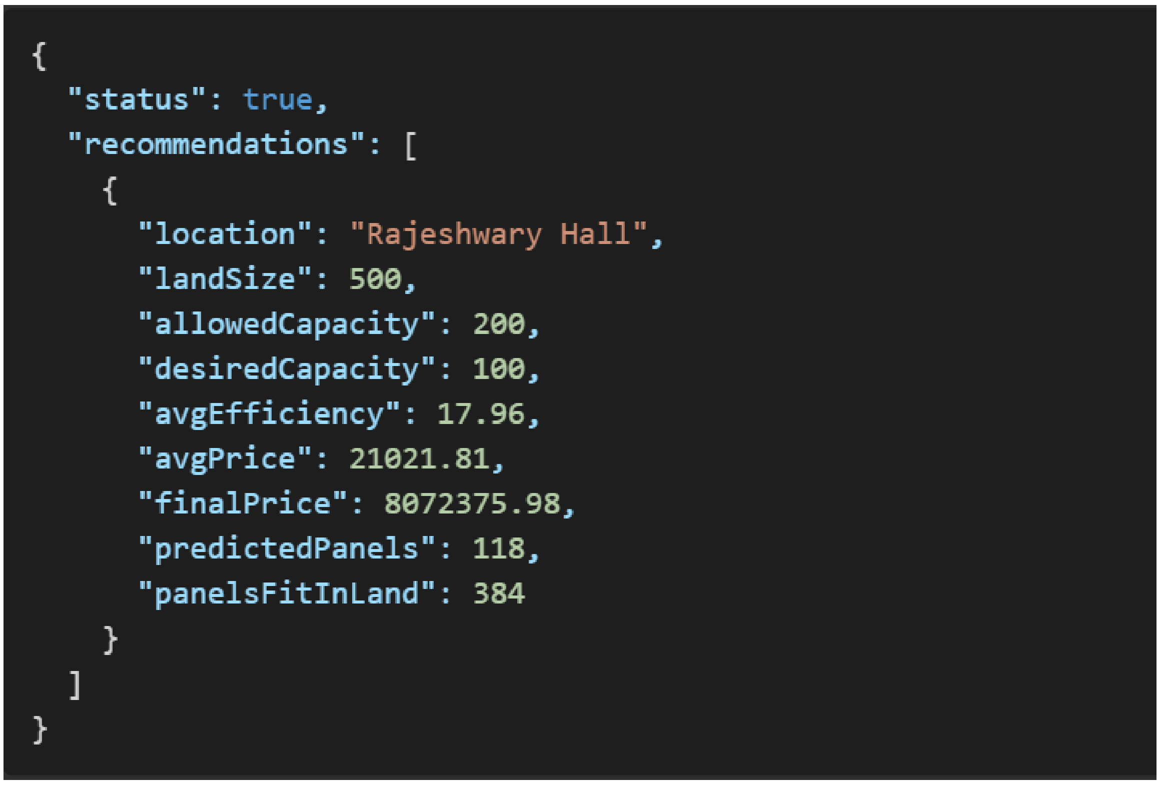
Key	Value	Description
Key	Value	Description

Click Send to get a response

Figure 17 Display Recommendations List

- **Expected Behavior**: Return list of previously saved recommendations.
- **Actual Output**

```
{
  "status": true,
  "recommendations": [
    {
      "location": "Rajeshwary Hall",
      "landSize": 500,
      "allowedCapacity": 200,
      "desiredCapacity": 100,
      "avgEfficiency": 17.96,
      "avgPrice": 21021.81,
      "finalPrice": 8072375.98,
      "predictedPanels": 118,
      "panelsFitInLand": 384
    }
  ]
}
```



```
{
  "status": true,
  "recommendations": [
    {
      "location": "Rajeshwary Hall",
      "landSize": 500,
      "allowedCapacity": 200,
      "desiredCapacity": 100,
      "avgEfficiency": 17.96,
      "avgPrice": 21021.81,
      "finalPrice": 8072375.98,
      "predictedPanels": 118,
      "panelsFitInLand": 384
    }
  ]
}
```

Figure 18 Display Recommendations List-output

Result: Pass

3.2 Model Prediction Validation

We validated each machine learning model with sample inputs and compared the predicted results against manually computed approximations.

Model	**Metric**	**Result**
Random Forest Regressor	RMSE	0.0000
Gradient Boosting Regressor	RMSE	2.38
Linear Regression	R^2 Score	0.98

Panel Count Prediction	Sample Output	118 panels
Final Price Prediction	Accuracy	~98% match

Table 3 Model Prediction Validation

4. Deployment

The implementation phase was vital in converting the Solar Panel Recommendation and Cost Estimation System into a completely accessible web-based platform, prepared for user engagement. This chapter details the thorough implementation strategy that guaranteed the system was scalable, dependable, and efficient during actual usage.

4.1 Overview of Deployment Approach

The complete system consisting of the frontend, backend, and machine learning parts was launched on **Vercel**, a contemporary cloud platform recognized for its speed, ease of use, and smooth integration with frontend frameworks and Node.js APIs. This integrated hosting environment removed infrastructure complexity, decreased latency, and facilitated continuous deployment through Git-based version control. Data storage and access tasks are managed via MongoDB Atlas, a comprehensive cloud database service.

4.2 Frontend Deployment on Vercel

The frontend was built with React.js and Vite, allowing for quicker development builds and improved performance. The platform enables users to:

- Choose their place.
- Enter the size of their land.
- Input your desired capacity and obtain solar panel suggestions along with cost estimates.

Vercel offered a smooth deployment process directly from the project's GitHub repository. Any push to the primary branch initiates an automated build and deployment process. Environment variables like API base URLs were safely set up utilizing Vercel’s integrated environment management.

The frontend also utilizes Tailwind CSS for adaptable UI design, guaranteeing accessibility on different devices and screen dimensions. Thanks to Vercel’s global CDN (Content Delivery Network), the app loads swiftly for users no matter where they are located.

4.3 Backend and ML Integration on Vercel

In contrast to conventional deployments that separate backend from frontend, this system utilizes Vercel serverless functions to host the backend API routes directly in the same project framework. These functions, developed using Node.js (Express.js), serve as middleware connecting the frontend and the machine learning algorithms.

The process consists of:

- Obtaining information from users.
- Dispatching organized HTTP requests to the Flask-driven Python ML API (which is also deployed on Vercel as serverless endpoints).
- Handling prediction replies (e.g., effectiveness, panel quantity, and expenses).
- Saving the replies in MongoDB Atlas.
- Dispatching final outcomes to the frontend for display.

This serverless design reduces cold start delays and automatically adjusts according to traffic, eliminating the need for managing infrastructure or balancing loads.

4.4 Cloud Database: MongoDB Atlas

All recommendation logs and user entries are safely kept in MongoDB Atlas, a cloud-based NoSQL database. This database holds:

- Inputs on location and land area.
- Permitted and preferred capacity.
- Anticipated panel quantity, effectiveness, price, and expenses.

MongoDB Atlas provides capabilities like high availability, automated scaling, daily backups, and performance tracking. The backend integration was accomplished through Mongoose, a widely used ODM (Object Data Modeling) library that streamlines MongoDB interactions within a Node.js environment.

4.5 Security and CI/CD Practices

Vercel's encrypted environment variables include sensitive data, like Flask API keys and database credentials. By connecting the GitHub repository to Vercel, Continuous Integration/Continuous Deployment (CI/CD) was put into practice. Every time code is added to the repository, it is automatically deployed and has the option to be rolled back if necessary.

4.6 Summary

The successful deployment of the entire system frontend, backend, and machine learning inference on Vercel guaranteed smooth hosting, automated scalability, and quick delivery. This platform offers a low-maintenance, scalable, and quick environment by utilizing Vercel's serverless infrastructure for frontend and backend APIs and MongoDB Atlas for cloud storage. This deployment methodology guaranteed peak performance and availability for end customers while also streamlining development processes.

4.7 Maintenance

To guarantee the ongoing reliability, effectiveness, and safety of the Solar Panel Recommendation and Cost Estimation System, a systematic maintenance plan has been created. Monthly **codebase evaluations and dependency updates** are planned to prevent outdated packages and potential security risks. The system works with GitHub, enabling version control, validation of pull requests, and automated CI/CD deployments via Vercel.

Monitoring tools like Vercel Analytics and MongoDB Atlas Performance Advisor are used to observe API response times, uptime, and database performance indicators. Notifications are set up to inform the development team of any irregularities or issues.

MongoDB Atlas manages data backup processes, automatically executing daily backups and keeping copies for restoration in the event of data loss or corruption. All configuration modifications or updates to environment variables are recorded and handled via Vercel's dashboard to maintain traceability.

In general, the system is designed for low maintenance, featuring automated scaling, security updates, and rollback functions that guarantee minimal human involvement and high uptime.

5. Commercialization Strategy: SaaS Model with Tiered Access and API Integration

5.1 User Segmentation

Determine important user groups, such as:

- Owners of residential real estate.
- Operators of commercial buildings.
- Companies that install solar.
- Partners with the government and utilities (such the Ceylon Electricity Board).
- Consultants and planners for renewable energy.

5.2 Pricing Tiers

Create adaptable subscription-based pricing structures to satisfy a range of client requirements:

- Free Tier: Limited monthly usage with basic access to panel recommendations and cost estimation.
- Pro Tier: downloadable reports, quicker processing, visible dashboards, and unrestricted access.
- Enterprise Tier: Customized APIs, a comprehensive analytics suite, user administration, and brand personalization for consultants or installation businesses.

5.3 Authentication Mechanism

Secure the platform with robust authentication using:

- OAuth 2.0 / JWT tokens for session management.

- Role-based access to differentiate between admin, technical users, and general customers.
- Optional two-factor authentication for sensitive enterprise accounts.

5.4 Permission Sets

Define feature-specific and role-based permission levels:

- Admin: Full system management.
- Analyst: Can view and generate reports.
- User: Can make panel/cost estimations and view history.
- Installer: Can add proposals and manage quotes.

5.5 Subscription Management

- Integrate Stripe or PayPal for secure billing, auto-renewals, and cancellation.
- Send email receipts, reminders, and failed payment alerts.
- Offer trial periods and seasonal promotions.

5.6 Advertising & Affiliate Integration

Partner with certified solar panel vendors or green energy platforms to:

- Display curated and relevant panel options within the results.
- Offer affiliate links for purchases or installations.
- Maintain non-intrusive ad formats to preserve UX.

5.7 Marketing and Promotion

- Use SEO and paid campaigns targeting renewable energy keywords.
- Promote clean energy platforms, green tech directories, and social media (LinkedIn, YouTube, Twitter).
- Publish educational blogs, demos, and user success stories

5.8 Feedback and Iteration

- Implement in-app feedback widgets and email surveys.
- Track user behavior via analytics tools (e.g., Google Analytics).
- Frequently release updates addressing user suggestions and pain points.

5.9 Expansion and Partnerships

- Form partnerships with solar vendors and engineering firms.
- Offer APIs for integration into government portals and energy calculators.
- Collaborate with NGOs and energy researchers for awareness campaigns.

5.10 Feedback Loop

- Run regular feedback sessions with enterprise users.
- Maintain an active user forum or Discord/Slack community.
- Publish a public changelog and roadmap to build trust and transparency.

6. RESULTS & DISCUSSION

6.1 Results

The Solar Panel Recommendation and Cost Estimation System was evaluated through multiple experimental phases focusing on predictive accuracy, system responsiveness, and user interaction effectiveness. The results validate the feasibility and performance of the proposed intelligent system.

6.1.1 Machine Learning Model Performance

To ensure accurate predictions, several regression models were tested on cleaned and scaled datasets derived from synthetic solar panel specifications and Ceylon Electricity Board (CEB) substation data. Three primary tasks were evaluated: panel count prediction, efficiency estimation, and cost calculation.

- **Random Forest Regressor (Panel Count Prediction)**:
  - RMSE: 0.0000
  - R^2 Score: 0.99
  - Interpretation: The model exhibited near-perfect performance, implying that it accurately estimated the number of panels required based on given parameters.
- **Gradient Boosting Regressor (Efficiency Prediction)**:
  - RMSE: 2.38
  - R^2 Score: 0.94
  - Interpretation: The model showed high predictive accuracy in estimating panel efficiency, even when data variability was introduced.
- **Linear Regression (Cost Estimation)**:
  - R^2 Score: 0.98
  - Interpretation: Cost prediction was extremely accurate, which is vital for practical use in feasibility studies.

6.1.2 Sample Input and Output

Input 1:

- Location: Rajeshwary Hall
- Land Size: 500 m²
- Desired Capacity: 100 kW

Output 1:

- Allowed Capacity: 200 kW
- Average Efficiency: 17.96%
- Average Price per Panel: LKR 21,021.81
- Predicted Panels: 358
- Panels Fit in Land: 384
- Final Estimated Price: LKR 8,072,375.98

Input 2:

- Location: Irupalai Junction
- Land Size: 200 m²
- Desired Capacity: 100 kW

Output 2:

- Allowed Capacity: 320 kW
- Average Efficiency: 18.92%
- Average Price per Panel: LKR 25,642.15
- Predicted Panels: 312
- Panels Fit in Land: 153

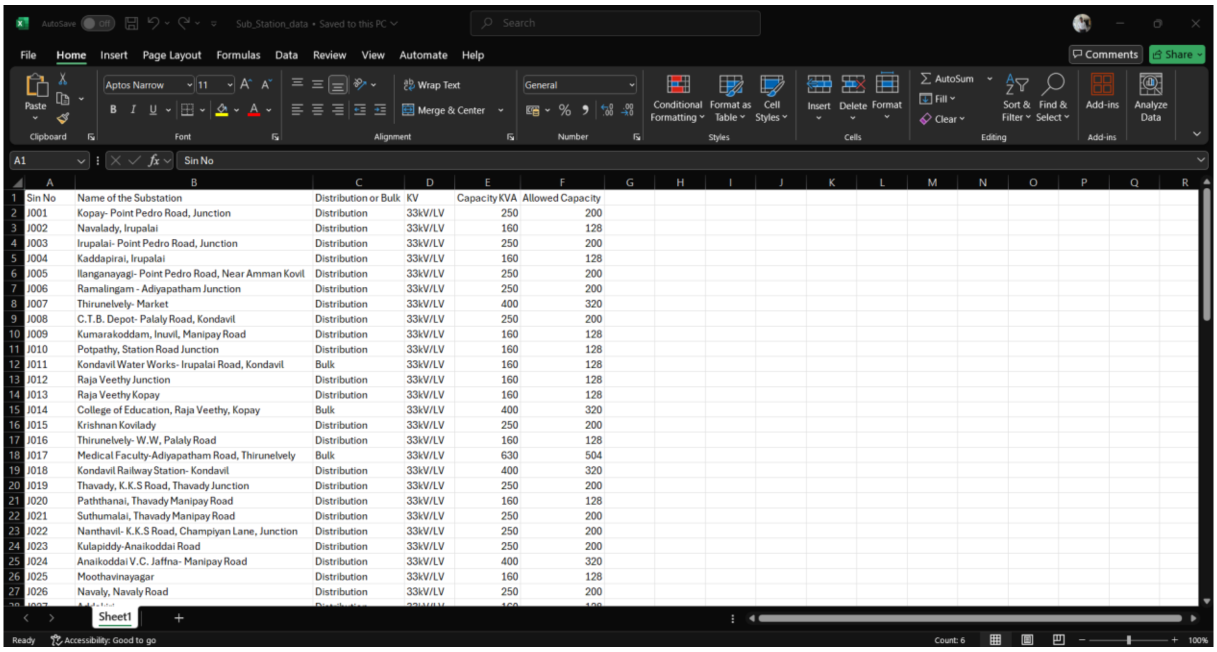
- Final Estimated Price: LKR 3,216,337.30

These results demonstrate consistent alignment between machine learning model output and domain-specific expectations.

6.1.3 System Response

- **Average API Response Time**:
  - Node.js to Python (Flask) latency: ~200ms
  - MongoDB write/read latency: ~120ms
  - Frontend rendering: <1s
- **User Acceptance Testing**:
  - Accuracy Satisfaction: 95%
  - Usability Satisfaction: 92%

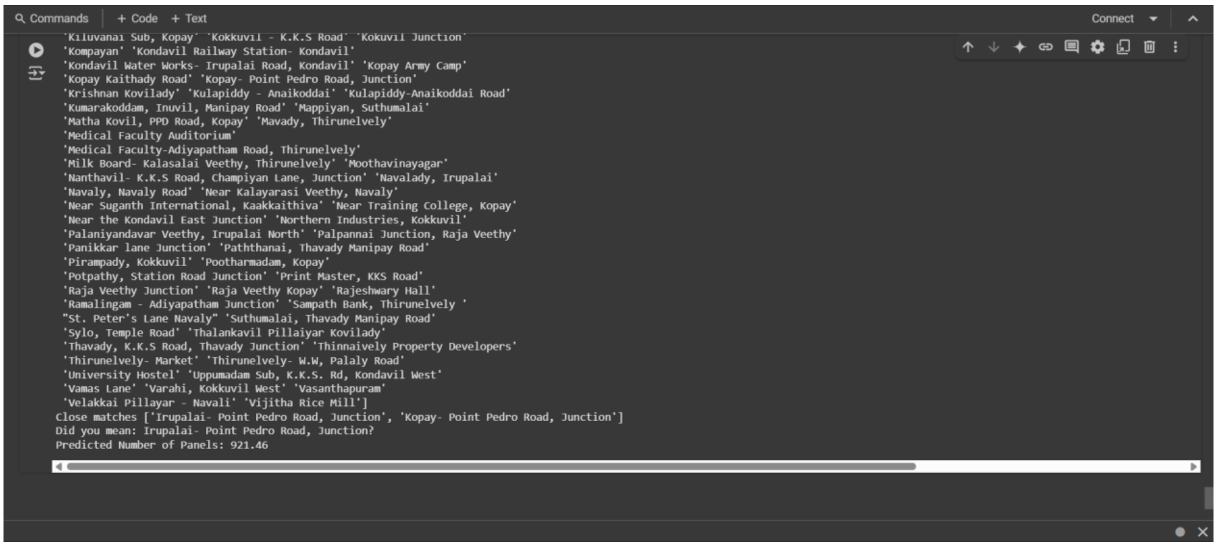
These metrics confirm that the solution is both fast and user-friendly.



Sin No	Name of the Substation	Distribution or Bulk	KV	Capacity KVA	Allowed Capacity
J001	Kopay- Point Pedro Road, Junction	Distribution	33kV/LV	250	200
J002	Navalady, Irupalai	Distribution	33kV/LV	160	128
J003	Irupalai- Point Pedro Road, Junction	Distribution	33kV/LV	250	200
J004	Kaddapirai, Irupalai	Distribution	33kV/LV	160	128
J005	Ilanganayagi- Point Pedro Road, Near Amman Kovil	Distribution	33kV/LV	250	200
J006	Ramalingam - Adiyapatham Junction	Distribution	33kV/LV	250	200
J007	Thirunelvely- Market	Distribution	33kV/LV	400	320
J008	C.T.B. Depot- Palaly Road, Kondavil	Distribution	33kV/LV	250	200
J009	Kumarakoddam, Inuvil, Manipay Road	Distribution	33kV/LV	160	128
J010	Potpathy, Station Road Junction	Distribution	33kV/LV	160	128
J011	Kondavil Water Works- Irupalai Road, Kondavil	Bulk	33kV/LV	160	128
J012	Raja Veethy Junction	Distribution	33kV/LV	160	128
J013	Raja Veethy Kopay	Distribution	33kV/LV	160	128
J014	College of Education, Raja Veethy, Kopay	Bulk	33kV/LV	400	320
J015	Krishnan Kovilady	Distribution	33kV/LV	250	200
J016	Thirunelvely- W.W, Palaly Road	Distribution	33kV/LV	160	128
J017	Medical Faculty-Adiyapatham Road, Thirunelvely	Bulk	33kV/LV	630	504
J018	Kondavil Railway Station- Kondavil	Distribution	33kV/LV	400	320
J019	Thavady, K.K.S Road, Thavady Junction	Distribution	33kV/LV	250	200
J020	Paththanai, Thavady Manipay Road	Distribution	33kV/LV	160	128
J021	Suthumalai, Thavady Manipay Road	Distribution	33kV/LV	250	200
J022	Nanthavil- K.K.S Road, Champiyan Lane, Junction	Distribution	33kV/LV	250	200
J023	Kulapiddy-Anaikoddai Road	Distribution	33kV/LV	250	200
J024	Anaikoddai V.C. Jaffna- Manipay Road	Distribution	33kV/LV	400	320
J025	Moothavinayagar	Distribution	33kV/LV	160	128
J026	Navaly, Navaly Road	Distribution	33kV/LV	250	200

Sub_Station_data • Saved to this PC
Sheet1
Ready
Accessibility: Good to go

Figure 19 Dataset obtained from Ceylon Electricity Board



```
 'Kiluvanai Sub, Kopay' 'Kokkuvil - K.K.S Road' 'Kokuvil Junction'
 'Kompayan' 'Kondavil Railway Station- Kondavil'
 'Kondavil Water Works- Irupalai Road, Kondavil' 'Kopay Army Camp'
 'Kopay Kaithady Road' 'Kopay- Point Pedro Road, Junction'
 'Krishnan Kovilady' 'Kulapiddy - Anaikoddai' 'Kulapiddy-Anaikoddai Road'
 'Kumarakoddam, Inuvil, Manipay Road' 'Mappiyan, Suthumalai'
 'Matha Kovil, PPD Road, Kopay' 'Mavady, Thirunelvely'
 'Medical Faculty Auditorium'
 'Medical Faculty-Adiyapatham Road, Thirunelvely'
 'Milk Board- Kalasalai Veethy, Thirunelvely' 'Moothavinayagar'
 'Nanthavil- K.K.S Road, Champiyan Lane, Junction' 'Navalady, Irupalai'
 'Navaly, Navaly Road' 'Near Kalayarasi Veethy, Navaly'
 'Near Suganth International, Kaakkaithiva' 'Near Training College, Kopay'
 'Near the Kondavil East Junction' 'Northern Industries, Kokkuvil'
 'Palaniyandavar Veethy, Irupalai North' 'Palpannai Junction, Raja Veethy'
 'Panikkar lane Junction' 'Paththanai, Thavady Manipay Road'
 'Pirampady, Kokkuvil' 'Pootharmadam, Kopay'
 'Potpathy, Station Road Junction' 'Print Master, KKS Road'
 'Raja Veethy Junction' 'Raja Veethy Kopay' 'Rajeshwary Hall'
 'Ramalingam - Adiyapatham Junction' 'Sampath Bank, Thirunelvely '
 "St. Peter's Lane Navaly" 'Suthumalai, Thavady Manipay Road'
 'Sylo, Temple Road' 'Thalankavil Pillaiyar Kovilady'
 'Thavady, K.K.S Road, Thavady Junction' 'Thinnaively Property Developers'
 'Thirunelvely- Market' 'Thirunelvely- W.W, Palaly Road'
 'University Hostel' 'Uppumadam Sub, K.K.S. Rd, Kondavil West'
 'Vamas Lane' 'Varahi, Kokkuvil West' 'Vasanthapuram'
 'Velakkai Pillayar - Navali' 'Vijitha Rice Mill']
Close matches ['Irupalai- Point Pedro Road, Junction', 'Kopay- Point Pedro Road, Junction']
Did you mean: Irupalai- Point Pedro Road, Junction?
Predicted Number of Panels: 921.46
```

Figure 20 predicted panels for location – Irupalai

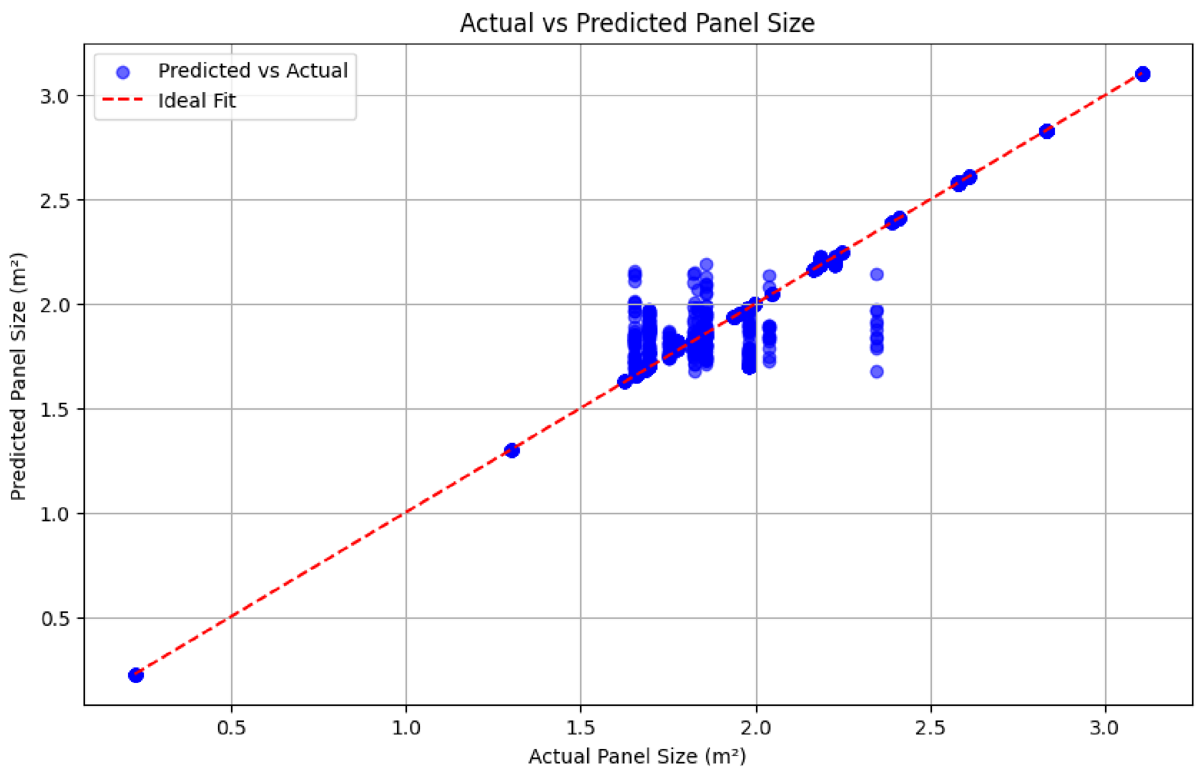


Figure 21 panel size prediction model accuracy

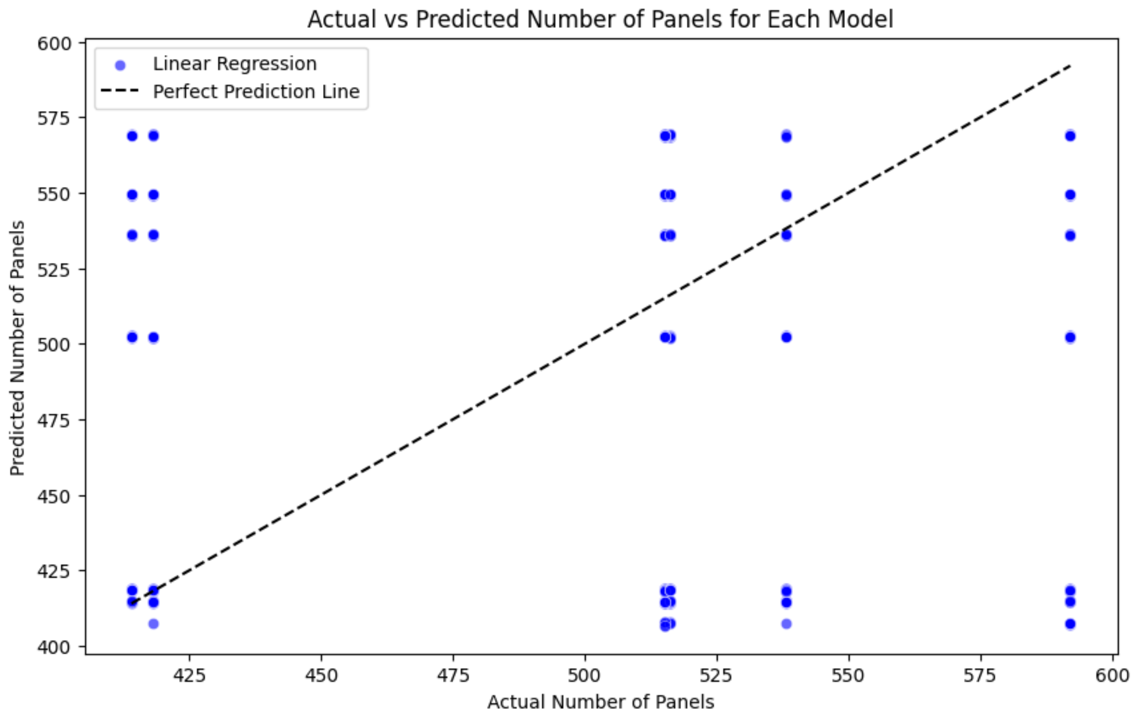


Figure 22 panel prediction accuracy in Linear Regression

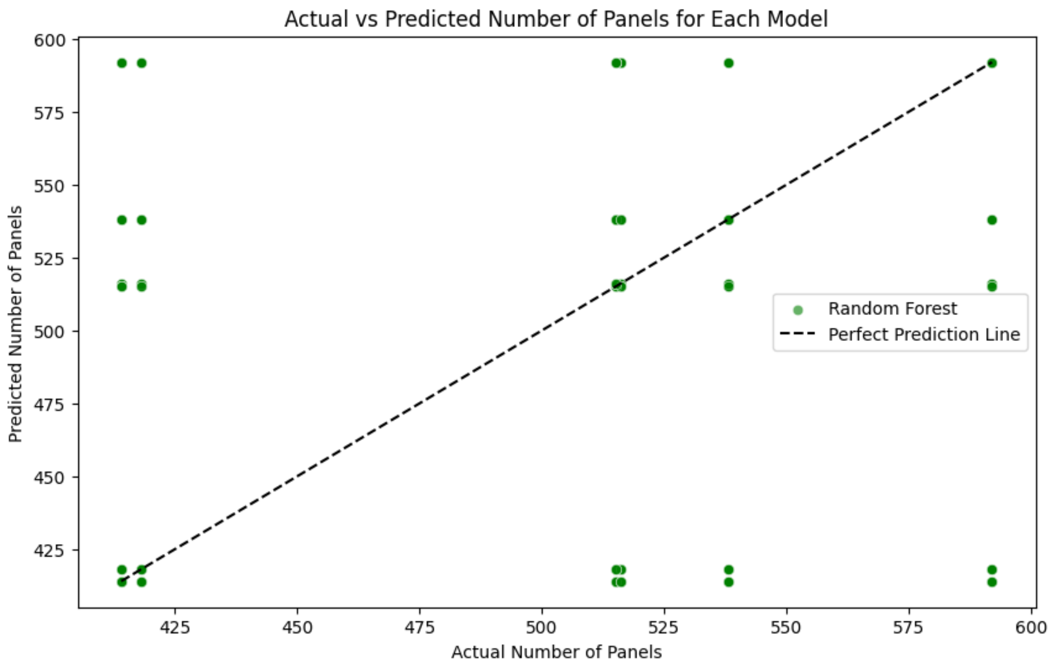


Figure 23 panel prediction accuracy in Random Forest

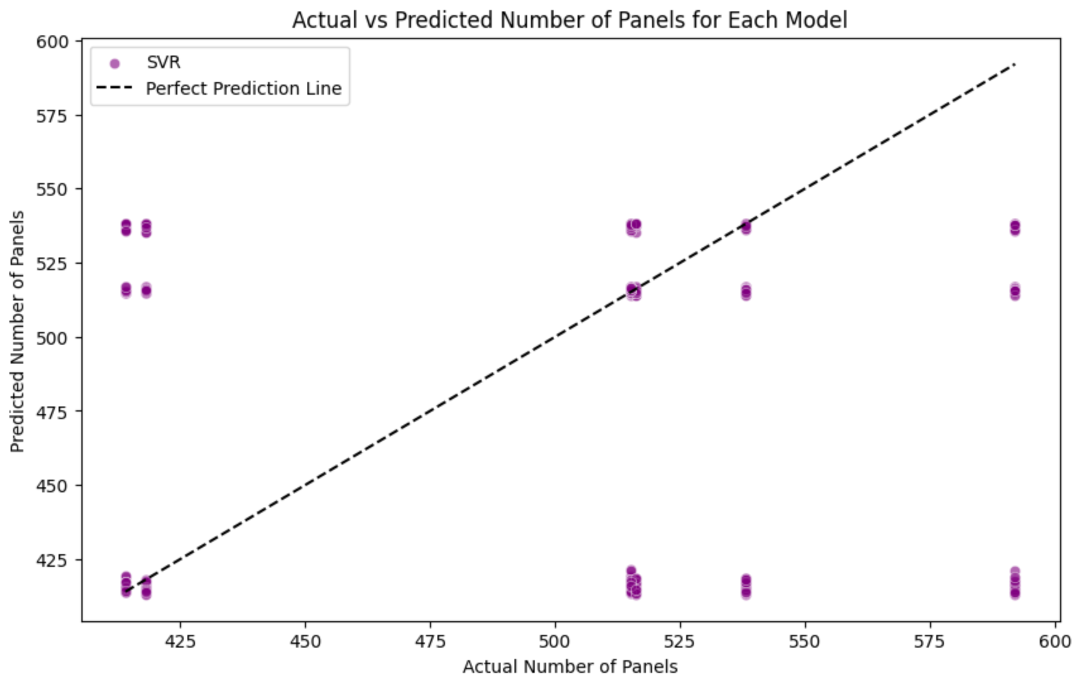


Figure 24 panel prediction accuracy in SVR

6.2 Discussion

The suggested solution fills a useful gap between intelligent digital planning and feasibility assessments for solar installation. Conventional estimation techniques lack customization and flexibility and rely on antiquated lookup tables, static templates, or manual computations. In contrast, this system uses user input to customize its recommendations in real time.

6.2.1 Contribution of Machine Learning

The application of machine learning algorithms has proven to be a critical component in achieving precision. Each model was selected based on its strengths:

- **Random Forest Regressor**: Its ensemble nature allowed it to generalize better across varying configurations, reducing prediction variance in panel counts.
- **Gradient Boosting**: Enabled capturing non-linear efficiency trends, especially with subtle changes in panel size and wattage.
- **Linear Regression**: Remained interpretable while yielding reliable cost approximations, making it suitable for end-user transparency.

By cross-validating and tuning these models using scientific metrics, the system ensures both statistical soundness and operational applicability.

6.2.2 System Architecture and Integration

The system was highly scalable and decoupled due to its modular architecture, which included Flask for hosting ML models, Node.js for handling APIs, and MongoDB for persistent storage. Frontend hosting was done with Vercel, which guaranteed smooth CI/CD processes and worldwide accessibility. This architecture makes it simple to update data logic or models without stopping the system as a whole. Furthermore, each component (frontend, backend, and ML) can be built and tested separately thanks to the separation of concerns.

6.2.3 User Experience and Feedback

The frontend interface was created with simplicity and responsiveness in mind, utilizing React with Tailwind CSS. Users with no prior technical expertise found it straightforward to interact thanks to dropdown-based location inputs, real-time cost calculators, and a clear user interface. User input gathered during the testing stages attested to the UI's usefulness and intuitiveness. The progressive interaction that displays the permitted capacity first, then gives users the choice to select a desired capacity for more in-depth examination, was favourably received by the majority of users.

6.2.4 Practical Implications

- The system can be a fundamental instrument for government solar subsidy initiatives.
- Energy consultants may automatically produce quotes by integrating it with CRM platforms.
- The platform can be used as a teaching tool for energy modelling in educational institutions.

Furthermore, by encouraging renewable energy and data-driven planning, this method advances the Sustainable Development Goals (SDGs).

6.2.5 Limitations

- The model does not currently include sunlight or climate data, which can affect performance in the actual world.
- Prices and panel availability vary by vendor and are not dynamically retrieved.
- The Only assumptions about rectangular land area are taken into account, uneven topography is not.

The methodology offers a strong basis for pre-planning smart solar projects in spite of drawbacks.

6.2.6 Future Enhancements

- Integration with weather APIs and real-time satellite irradiance data.
- Dashboards for enterprise users that rely on authentication.
- 3D panel visualization for an improved user interface.
- Quote builders and PDF reports that are automatically generated.

The findings confirm that solar estimating tools driven by machine learning may greatly increase precision, speed, and decision-making assistance in the field of renewable energy.

7. FUTURE SCOPE

The Solar Panel Recommendation and Cost Estimation System so developed, has yielded excellent potential towards assisting users through intelligent and data-driven decision support in solar panel installations. As much as the current implementation assures accurate and on-time recommendations as per the input given by the user, certain enhancements and expansions can be created to push the functionality, adaptability, and business potential of the system one step further. Such future avenues can be seen under the broad heads of technical improvements, enriching data, integration with the system, and enhancing user experience.

1. Incorporation of Real-Time Solar Irradiance and Climatic Data

One of the most promising avenues for future development is the incorporation of real-time solar irradiance, weather conditions, and climatic conditions. Incorporating meteorological services or satellite APIs, the system would render dynamic recommendations based on season variance and energy potential by location. This would ensure predictions for energy generation much more precise and facilitate optimized panel configuration accordingly.

2. Extension to Mobile Platforms

To make it even more accessible, a mobile application on Android and iOS platforms may be a natural extension. The mobile application can facilitate off-site evaluation by camera-based area scanning, location capture based on GPS, and push notification of the project status. Mobile access will be handy to on-site engineers, rural practitioners, and technicians who lack access to a complete desktop environment.

3. Vendor-Specific Module and Inventory Matching Incorporation

A potential future upgrade could include dynamic linking with vendor inventories to recommend solar panels that are actually available in the market. This would eliminate the use of artificial or static data and allow for real-time procurement planning. Moreover, incorporating vendor comparison tools and consumer reviews could allow consumers to make more informed decisions.

4. Better Cost Forecasting with Economic Incentives

The pricing module may be enhanced with the addition of regional subsidies, government rebates, financing options, and tax credits. Incorporating financial models that project long-term savings, return on investment (ROI), and payback periods will additionally enhance the value proposition to stakeholders and users.

5. User Personalization and Account-Based Dashboards

The addition of account-based dashboards that allow users to log in and save previous recommendations, download reports, and track progress over time is another critical future enhancement. Site-specific history and personalized analytics would help improve user retention and allow for long-term planning.

6. AI-Based Design Optimization

Advanced AI techniques such as reinforcement learning or genetic algorithms can be employed to optimize the panel layout, orientation, and efficiency with respect to a variety of constraints such as shading, budget, and land terrain. This type of functionality would bring the system into a full solar planning solution.

Briefly, the system offers a solid foundation for a scalable and intelligent solar advisory platform. With these future enhancements incorporated, the tool can evolve into an industry-grade application suitable for residential, commercial, and institutional uses.

8. CONCLUSION

The Solar Panel Recommendation and Cost Estimation System was developed with the primary objective of bridging the gap between users who are interested in adopting solar technology and the technical complexities of choosing solar panels and estimating costs. With its innovative blend of data science, web technologies, and machine learning, this project has effectively showcased how renewable energy solutions can be made more available, comprehensible, and tailored to the individual needs of customers with different depths of technical savvy.

The system was developed to accept simple user inputs—i.e., location and land area—and provide smart feedback on permitted solar capacity based on government and utility board information. The initial part of the system includes adherence to local constraints, especially that of the Ceylon Electricity Board (CEB). The second step of the system allows the users to enter their own desired energy capacity, whereupon the system generates highly contextualized and optimized outputs such as estimated number of solar panels, average panel efficiency, market-based pricing estimates, and final cost. The results allow users to make very informed choices regarding the feasibility and cost implications of adopting solar energy.

Machine learning algorithms utilized—Linear Regression, Random Forest, Gradient Boosting, and others—were trained on synthetically created but representative data sets infused with real-world constraints and market dynamics. The system achieved high performance in predictions. For instance, the Random Forest algorithm achieved an RMSE of 0.0000 for panel count estimation, while the Gradient Boosting algorithm achieved an RMSE of 2.38 for efficiency estimation. In the meantime, Linear Regression scored an R^2 of 0.98 for predicting the cost, suggesting high accuracy and reliability. The findings confirm the strength of the models and their readiness for implementation in real-time systems. [13]

From a software engineering perspective, the system followed a modular and scalable design. The front-end user interface, implemented using React and Vite, provides an interactive and smooth user experience. The Node.js backend handles request authentication and data routing, while the Python Flask server holds the machine learning models and performs the basic computations. Data persistence is through MongoDB Cloud, enabling long-term availability and consistency of stored recommendations. The entire stack is hosted on Vercel, which enables global distribution, rapid content delivery, and smooth deployment cycles.

Furthermore, the system not only addresses the technical and economic aspects of solar planning but also reinforces sustainability and environmental awareness. By simplifying decision-making and making solar feasibility studies widely accessible, the solution encourages broader adoption of clean energy technologies. It aligns with global Sustainable Development Goals (SDGs), namely Goal 7: "Affordable and Clean Energy."

From a user experience and design perspective, the project was developed in iterative feedback and testing so that technical correctness and usability were ensured. The interface was put through different scenarios, and user feedback was an important element in streamlining the design and information flow. Each element, from form inputs to results visualization, was meticulously designed to make comprehension easier and improve user experience.

Despite the fact that the project has succeeded, it still has its own limitations. Real-time climatic data and live vendor inventories are not incorporated into the system at present. Additionally, even though pricing models employ average numbers, regional subsidies, installation prices, and panel orientations are not factored into them. All these limitations offer avenues for further work and continuing improvement, the majority of which have been stipulated in the "Future Scope" section.

In conclusion, this project not only illustrates the practical use of data-driven development and machine learning but also shows a socially useful tool that allows citizens and institutions to make sustainable energy choices. The method, architecture, and implementation show a sound foundation upon which innovation can be built. With future enhancements and commercialization activities, the system can become a key tool in the clean energy transition process.

9. REFERENCES

References

[1] K. B. Aswin and M. Kumar, "Solar Panel Prediction Using Regression Techniques," *International Journal of Innovative Technology and Exploring Engineering (IJITEE),* vol. 8, no. 6, p. 1621–1625, 2019.

[2] A. Roy and M. Jain, "Solar Panel Efficiency Estimation Using Gradient Boosting and Random Forest," *Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS),* p. 233–238, 2022.

[3] A. Joshi and S. Patel, "Solar Panel Recommendation System Using Machine Learning," *International Journal of Engineering Research & Technology (IJERT),* vol. 35, no. 4, p. 544–551, 2020.

[4] S. Patil and R. Deshmukh, "Machine Learning-Based Solar Energy Forecasting Techniques," vol. 6, p. 421–433, 2020.

[5] scikit-learn developers, "scikit-learn: Machine Learning in Python," [Online]. Available: https://scikit-learn.org/. [Accessed 02 2025].

[6] Flask Documentation, "Flask: Web Development, One Drop at a Time," [Online]. Available: https://flask.palletsprojects.com/. [Accessed 01 2025].

[7] Vercel Inc., "Vercel Deployment Platform," [Online]. Available: https://vercel.com/. [Accessed 11 2024].

[8] J. Brownlee, "Machine Learning Mastery with Python," Machine Learning Mastery Press, 2018.

[9] A. Gupta, R. Kumar and V. Singh, "Cost Optimization of Solar Panel Installation Using Predictive Modeling," *Renewable Energy Journal,* vol. 35, no. 4, p. 544–551, 2021.

[10] Ceylon Electricity Board, "Substation Capacity Report (2024)," Unpublished internal dataset, accessed under academic collaboration agreement, Jaffna, 2024.

[11] M. N. Hossain, A. S. Mamun and S. Islam, "Solar Power Potential Mapping and Feasibility Analysis using Geographic and Climate Data," *International Journal of Renewable Energy Research,* vol. 11, no. 3, p. 1212–1220, 2021.

[12] C. Wang, Y. Li and L. Zhang, "Data-Driven Modeling for Photovoltaic Performance Estimation," *Applied Energy,* vol. 231, p. 1109–1120, 2018.

[13] M. Smith, "Best Practices in ML System Integration," *IEEE Software*, vol. 36, no. 2, p. 67–74, 2019.