A

Major Project Report

On

SOLAR ENERGY FORECASTING USING MACHINE LEARNING TECHNIQUES FOR ENHANCED GRID STABILITY

Submitted to CMREC (UGC Autonomous), Affiliated to JNTUH

In Partial Fulfilment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

(Artificial Intelligence & Machine Learning)

Submitted By

K.HARINI	**(228R1A66F3)**
K.KALYAN	**(228R1A66F6)**
M.VISHNU	**(228R1A66G3)**
V. PHANEENDRA	**(228R1A66K0)**

Under the Esteemed guidance of

Dr. MADHAVI PINGILI

Professor & HOD

Department of CSE (AI & ML)

Department of Computer Science & Engineering (AI & ML)

CMR ENGINEERING COLLEGE

(UGC AUTONOMOUS)

(Accredited by NAAC & NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad, Kandlakoya, Medchal Road, R. R. Dist. Hyderabad-501 401)

(2025 – 2026)

CMR ENGINEERING COLLEGE

(UGC AUTONOMOUS)

(Accredited by NAAC & NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad, Kandlakoya, Medchal Road, R. R. Dist. Hyderabad-501 401)

Department of Computer Science & Engineering (AI & ML)

CERTIFICATE

This is to certify that the Major project entitled **"SOLAR ENERGY FORECASTING USING MACHINE LEARNING TECHNIQUES FOR ENHANCED GRID STABILITY"** is a bonafide work carried out by

K.HARINI	**(228R1A66F3)**
K.KALYAN	**(228R1A66F6)**
M.VISHNU	**(228R1A66G3)**
V. PHANEENDRA	**(228R1A66K0)**

in partial fulfilment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING (AI & ML)** from CMR Engineering College, under our guidance and supervision

The results presented in this Major project have been verified and are found to be satisfactory. The results embodied in this Major project have not been submitted to any other university for the award of any other degree or diploma

Internal Guide	**Major Project Coordinator**	**Head of the Department**
Dr. Madhavi Pingili	Mr. G. Venkateswarlu	Dr. Madhavi Pingili
Professor & HOD	Assistant Professor	Professor & HOD
Department of CSE(AI&ML)	Department of CSE(AI&ML)	Department of CSE(AI&ML)

External Examiner : _____

DECLARATION

This is to certify that the work reported in the present Major project entitled **"SOLAR ENERGY FORECASTING USING MACHINE LEARNING TECHNIQUES FOR ENHANCED GRID STABILITY"** is a record of bonafide work done by us in the Department of Computer Science and Engineering (AI & ML), CMR Engineering College. The reports are based on the Major project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the Major project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

K.HARINI	**(228R1A66F3)**
K.KALYAN	**(228R1A66F6)**
M.VISHNU	**(228R1A66G3)**
V. PHANEENDRA	**(228R1A66K0)**

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. Madhavi Pingili,** Professor & HOD, **Department of CSE (AI & ML), CMR Engineering College** for their constant support.

We are extremely thankful to **Dr. Madhavi Pingili,** Professor & HOD, Internal Guide, Department of CSE (AI & ML), for her constant guidance, encouragement and moral support throughout the Major project.

We will be failing in duty if we do not acknowledge with grateful thanks the authors of the references and other literature referred to in this Major project.

We thank **Mr. G. Venkateswarlu**, Major Project Coordinator, for his constant support in carrying out the Major project activities and reviews.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Major project successfully in time.

Finally, We are very much thankful to our parents who guided us every step.

K.HARINI **(228R1A66F3)**

K.KALYAN **(228R1A66F6)**

M.VISHNU **(228R1A66G3)**

V. PHANEENDRA **(228R1A66K0)**

CONTENTS

TOPIC	PAGE NO
ABSTRACT	**I**
LIST OF FIGURES	**II**
LIST OF TABLES	**III**
1. INTRODUCTION	**1**
1.1. Introduction and Objectives	1
1.2. Project Objectives	2
1.3. Purpose of the project	3
1.4 Problem Statement	4
1.5. Existing System and Disadvantages	5
1.6 Proposed System and Advantages	6
1.7 Input and Output Design	7
2. LITERATURE SURVEY	**11**
3. SOFTWARE REQUIREMENT ANALYSIS	**15**
3.1. Modules and their Functionalities	15
3.2. Functional Requirements	16
3.3. Non-Functional Requirements	17
3.4. Feasibility Study	18
4. SYSTEM SPECIFICATIONS	**20**
4.1. Software requirements	20
4.2. Hardware requirements	21
5. SOFTWARE DESIGN	**22**
5.1. System Architecture	22
5.2. Dataflow Diagram	23
5.3. UML Diagrams	25

6. CODING AND IMPLEMENTATION **38**

6.1. Source Code 38

6.2. Implementation 57

7. SYSTEM TESTING **63**

7.1. Types of System Testing 64

7.2. Testing Strategies 67

7.3. Sample Testcases 72

8. RESULTS **80**

9. CONCLUSION **86**

10. FUTURE ENHANCEMENTS **88**

REFERENCES **89**

ABSTRACT

The increasing integration of solar photovoltaic (PV) systems into modern power grids has created a strong need for accurate short-term forecasting due to the intermittent and highly variable nature of solar energy. Traditional forecasting methods, such as numerical weather prediction and radiation transfer models, are computation-heavy, dependent on precise atmospheric data, and often unsuitable for real-time applications.

To overcome these limitations, this project proposes a machine-learning-based solar forecasting framework that intelligently combines meteorological parameters—temperature, humidity, cloud cover, wind speed, and solar irradiance—with solar geometric features such as zenith angle, azimuth, and angle of incidence. This fusion of diverse inputs enables the system to capture nonlinear patterns in solar irradiance and significantly improves prediction accuracy. The methodology follows a structured pipeline starting with data collection from sensors, historical PV power records, and solar position calculations, followed by preprocessing techniques like cleaning, imputation, and time-alignment to ensure consistent and reliable datasets. Feature engineering is employed to derive additional predictive factors, while feature selection identifies the most influential attributes affecting PV output. Multiple machine learning models, including Random Forest, Decision Tree Regression, Support Vector Regression, and Gradient Boosting Regressor (GBR), are trained under uniform evaluation conditions. Hyperparameter optimisation and k-fold cross-validation improve model robustness and reduce overfitting. Among these models, the Gradient Boosting Regressor demonstrates the highest accuracy, lowest error metrics, and strongest ability to handle nonlinear and time-varying relationships in solar data. The system delivers dependable short-term solar power predictions that support grid stability, load balancing, and energy dispatch planning. By enhancing forecasting reliability, the proposed system reduces dependency on backup sources, increases operational efficiency, and improves the economic viability of renewable energy integration. Overall, the proposed machine-learning-driven solar forecasting framework provides a practical, scalable, and intelligent solution for managing solar PV variability and advancing smart grid performance.

Keywords: Solar Forecasting, Machine Learning, Gradient Boosting Regressor, Solar Geometry, Renewable Energy, Smart Grid.

LIST OF FIGURES

S.NO	FIGURE NO	DESCRIPTION	PAGE NO
1	1.6.1	Block diagram of proposed system	7
2	5.1	System Architecture	22
3	5.2	Data Flow Diagram	24
4	5.3.1	Use Case Diagram	28
5	5.3.2	Class Diagram	29
6	5.3.3	Sequence Diagram	33
7	5.3.4	Activity Diagram	35
8	7.3.1	User Registration	73
9	7.3.2	User Login	74
10	7.3.3	Solar Energy Prediction	75
11	7.3.4	Prediction History	76
12	7.3.5	Admin User Management	77
13	7.3.6	Invalid Username	78
14	7.3.7	Password Mismatch Validation	79
15	8.1	Home page– Solar Energy Predictor	80
16	8.2	User Login	81
17	8.3	Profile View	82
18	8.4	Data Visualization	83
19	8.5	Admin User Management	84
20	8.6	Analytics View	85

LIST OF TABLES

S.NO	TABLE NO	DESCRIPTION	PAGE NO
1	2	Literature Review Summary	14
2	7.3	Test Cases	72

1. INTRODUCTION

1.1 Introduction

The integration of solar energy into modern power grids has gained significant momentum in recent years due to its environmental sustainability and declining installation costs **[7], [13]**. However, the intermittent and uncertain nature of solar power generation, primarily influenced by dynamic weather conditions such as cloud cover, temperature, and atmospheric variations **[3], [12]**, creates major challenges for maintaining grid stability and reliability **[4], [9]**. Accurate solar energy forecasting is therefore essential to ensure efficient grid operation, optimal energy resource allocation, and improved economic viability of renewable energy systems **[5], [10]**.

Traditional forecasting approaches, including statistical and physical models such as numerical weather prediction and radiation transfer models **[12], [13]**, rely heavily on atmospheric equations and predefined parameters. Although these models provide theoretical insights, they often struggle to capture the nonlinear and stochastic behaviour of solar radiation and require high computational resources, limiting their effectiveness in real-time applications. These limitations have led researchers to explore data-driven approaches, particularly machine learning (ML) and deep learning (DL) techniques, which offer improved adaptability and predictive accuracy **[8]–[12]**.

Machine learning algorithms such as Support Vector Machines (SVMs), Random Forests, and Artificial Neural Networks (ANNs), along with deep learning architectures like Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTMs) networks, have demonstrated strong capabilities in modelling complex relationships between meteorological variables and solar power generation **[10], [14], [15]**. These models leverage historical datasets to identify hidden patterns and provide accurate predictions under varying environmental conditions. Training strategies such as backpropagation, adaptive optimisers, and regularisation techniques further enhance model performance and generalisation.

Despite these advancements, several existing studies overlook important solar geometric parameters such as solar zenith angle, azimuth angle, and angle of incidence, or fail to evaluate models using standardised training and validation frameworks **[1], [6]**. This limits the robustness and scalability of forecasting systems in real-world grid environments. Therefore, there is a need for a

comprehensive and unified forecasting pipeline that integrates meteorological data, solar geometry, and real-time weather information to improve prediction accuracy and grid stability.

The proposed system aims to provide a scalable and interpretable solar forecasting pipeline suitable for smart grid applications. By analysing model performance, feature importance, and hyperparameter configurations, this study offers practical insights for researchers, energy system operators, and policymakers to improve renewable energy integration and ensure reliable grid management. Ultimately, the proposed approach contributes to the development of efficient, sustainable, and economically viable solar energy forecasting systems.

1.2 Project Objectives

The primary objective of this project is to design and develop a machine learning–based solar energy forecasting system capable of accurately predicting short-term solar photovoltaic (PV) power generation using meteorological and solar geometric parameters **[1], [2], [3]**. By leveraging advanced machine learning algorithms such as Random Forest, Support Vector Regression, and Gradient Boosting, the system aims to analyse historical solar radiation and weather data to generate reliable power forecasts **[7], [8], [9]**. This approach reduces dependency on traditional statistical and physical models and provides a more adaptive and scalable solution suitable for real-world smart grid environments **[4], [5]**. The proposed system transforms raw solar and weather datasets into meaningful predictive insights, enabling efficient energy planning and operational decision-making **[10], [11]**.

Another important objective of this project is to compute and present performance metrics that support smart grid stability and efficient energy management. The system focuses on extracting key forecasting indicators such as prediction accuracy, error rates, and model performance using evaluation metrics like Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R^2 score **[8], [12]**. Additionally, the forecasting results are visualised through graphs and comparative analysis, allowing users to easily interpret solar power trends and generation patterns **[6], [15]**. By integrating real-time meteorological data and feature engineering techniques, the proposed system ensures reliable solar energy predictions, which ultimately contribute to load balancing, resource allocation, and sustainable grid operation **[3], [16]**.

Objectives:

The develop a machine learning–based forecasting model that accurately predicts short-term solar photovoltaic (PV) power generation using meteorological and solar geometric data.

- To collect and preprocess real-world solar and weather datasets by handling missing values, normalisation, and feature extraction to improve model performance.
- To apply feature engineering techniques to identify important parameters such as solar radiation, temperature, humidity, cloud cover, and solar angles that influence solar energy generation.
- To implement and compare multiple machine learning algorithms such as Random Forest, Support Vector Regression (SVR), Gradient Boosting, and Artificial Neural Networks to determine the most accurate forecasting model.
- To optimise model performance using hyperparameter tuning, training-validation techniques, and evaluation metrics like MAE, RMSE, and R^2 score.
- To integrate real-time weather data into the forecasting system to improve prediction accuracy under dynamic environmental conditions.
- To support smart grid stability and energy management by providing reliable solar power forecasts for efficient load balancing, resource allocation, and sustainable energy utilisation.

1.3 Purpose of the Project

The purpose of this project is to develop an intelligent and reliable machine learning–based framework for accurately forecasting solar photovoltaic (PV) power generation **[1], [2], [4]**. The system aims to integrate meteorological parameters and solar geometric data into a structured preprocessing and analysis pipeline to generate precise short-term solar energy predictions **[3], [6]**. By transforming raw solar and weather data into meaningful forecasting outputs, the project supports efficient energy planning and improves the stability of modern power grids **[5], [7]**. This approach enables better utilisation of renewable energy resources and facilitates smooth integration of solar power into smart grid systems **[8], [9]**.

Another important purpose of this project is to enhance the accuracy and efficiency of solar energy forecasting by applying advanced machine learning algorithms and feature engineering techniques **[10], [11]**. Traditional forecasting models often struggle to handle nonlinear relationships and dynamic weather conditions, which leads to inaccurate predictions **[12], [13]**. By using machine

learning methods such as Random Forest, Support Vector Regression, and Gradient Boosting, the proposed system can learn complex patterns from historical solar and meteorological datasets and provide reliable forecasts under varying environmental conditions **[7], [14], [15]**. This improves prediction performance and ensures adaptability in real-time energy systems.

Furthermore, the project aims to support grid stability and energy management by providing dependable solar power forecasts for load balancing and resource allocation **[4], [8]**. The system evaluates model performance using standard metrics such as MAE, RMSE, and R^2 **score** and presents results through visual analysis for better interpretation **[10], [16]**. By enabling accurate forecasting and efficient decision-making, the proposed framework contributes to sustainable energy utilisation and the development of intelligent smart grid infrastructure, ultimately supporting the transition toward clean and renewable energy systems **[1], [5]**.

1.4 Problem Statement

Traditional solar energy forecasting systems mainly rely on statistical models and basic machine learning techniques, which are not effective in handling the intermittent and unpredictable nature of solar photovoltaic (PV) power generation **[10], [12]**. Variations in weather conditions such as solar irradiance, temperature, humidity, and cloud cover directly affect solar output, making accurate forecasting a challenging task **[2], [3]**. Although some existing methods perform well in controlled environments, they often fail to deliver reliable predictions in real-world grid-connected solar systems due to dynamic environmental conditions and fluctuating energy demand **[4], [7]**.

These systems require extensive preprocessing, manual feature engineering, and structured datasets to achieve reasonable accuracy, which increases system complexity and processing time **[6], [11]**. Moreover, many forecasting models focus only on prediction accuracy and do not consider grid stability and energy management requirements **[8], [9]**. Rapid changes in solar power generation can lead to power imbalance, inefficient load distribution, and difficulty in maintaining stable grid operations **[5], [13]**. As a result, existing approaches are not fully capable of supporting modern smart grid environments **[1], [4]**.

In addition, several advanced solutions depend on costly infrastructure such as energy storage systems, distributed energy resources, and complex communication networks, which limits their scalability and practical implementation **[14], [15]**. Therefore, there is a need for an intelligent and efficient solar energy forecasting system that can accurately predict photovoltaic power generation

using machine learning techniques, integrate meteorological data effectively, and support grid stability and energy management in a cost-effective and scalable manner .

1.5 Existing System and Disadvantages

The existing solar energy forecasting system is primarily based on Numerical Weather Prediction (NWP) and radiation transfer models, which are widely used for estimating solar irradiance and photovoltaic (PV) power generation **[6], [7]**. In this approach, the system focuses on analysing atmospheric and meteorological parameters such as temperature, humidity, cloud cover, and solar radiation to predict energy output **[3], [4]**. These models rely on physical equations and weather simulations to estimate solar power generation over time **[6]**. Although this method provides theoretical accuracy and scientific understanding of atmospheric conditions, it performs effectively only in controlled environments where accurate and complete weather data is available and environmental conditions remain stable **[7], [12]**.

However, despite its effectiveness in theoretical and research-based environments, the traditional solar forecasting system faces significant challenges when applied to real-world smart grid scenarios **[1], [4]**. It heavily depends on precise weather inputs and complex atmospheric calculations, which become difficult to manage under rapidly changing environmental conditions such as cloud fluctuations, seasonal variations, and temperature changes **[3], [5]**. Additionally, many existing machine learning approaches fail to incorporate important solar geometric parameters such as azimuth angle, zenith angle, and angle of incidence, which limits prediction accuracy **[2], [16]**. As a result, the existing system is not well-suited for real-time solar forecasting and fails to provide reliable and scalable predictions required for efficient grid stability and energy management **[8], [10]**.

Disadvantages

- High computational cost: NWP and physical models require significant processing power, making them unsuitable for real-time forecasting.
- Dependence on precise input data: Performance drops sharply if weather data is noisy, incomplete, or inaccurate.
- Poor generalisation: Models trained in one region struggle to adapt to different climatic conditions.
- Limited use of solar geometry: Many systems ignore azimuth, zenith, and incidence angles, reducing prediction accuracy.

- Instability in ML models: Without proper tuning and feature engineering, ML approaches may overfit and show inconsistent performance.

1.6 Proposed system and Advantages

The proposed system is designed using a machine learning–based solar forecasting framework to overcome the limitations of traditional Numerical Weather Prediction (NWP) and basic machine learning models **[1], [4]**. As illustrated in the system architecture, the workflow begins with the Data Input Module, where historical solar and meteorological data such as temperature, humidity, cloud cover, wind speed, and solar radiation are collected and uploaded into the system **[5]**. These datasets are then processed in the Data Preprocessing Module, where missing values are handled, data is normalized, and solar geometric parameters such as zenith angle, azimuth angle, and angle of incidence are integrated. Unlike traditional forecasting methods, this structured preprocessing pipeline enables the system to work efficiently with real-world solar datasets and improves prediction reliability under varying environmental conditions **[6]**.

Following preprocessing, the system utilizes the Machine Learning Model Module, where multiple algorithms such as Random Forest, Support Vector Regression, Decision Tree Regression, and Gradient Boosting Regressor are applied to predict solar photovoltaic (PV) power generation **[2]**. The models are trained and validated using cross-validation and hyperparameter optimization techniques to ensure high accuracy and generalization. Among the evaluated models, the Gradient Boosting Regressor provides the most accurate and robust predictions by effectively capturing nonlinear relationships between meteorological variables and solar power output **[3]**. The trained model then generates short-term solar energy forecasts, which are further analyzed in the Performance Evaluation Module using metrics such as MAE, RMSE, and R^2 score to measure prediction accuracy and reliability .

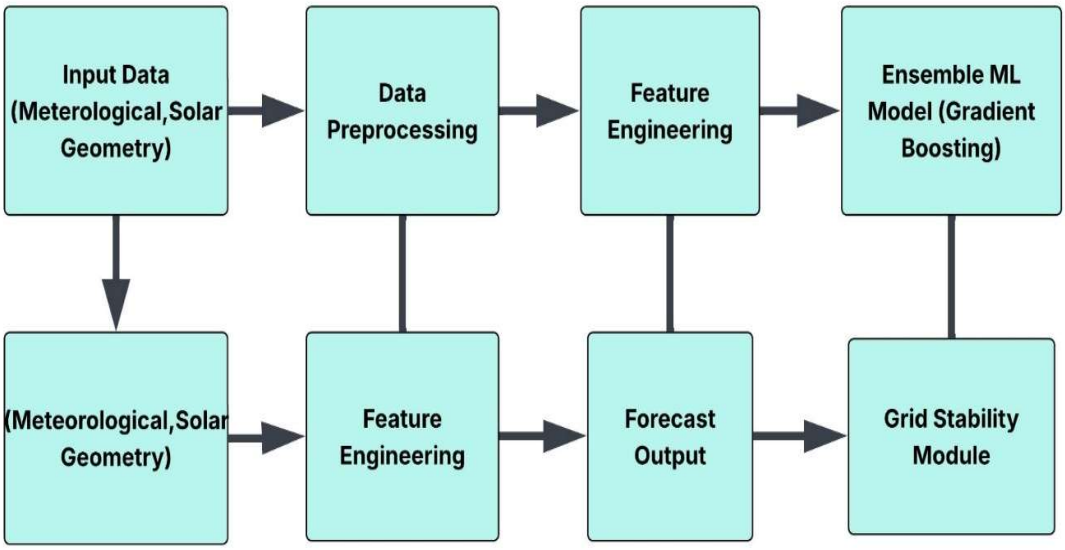

Figure 1.6.1: Block diagram of proposed system

Advantages:

- High computational cost: NWP and physical models require significant processing power, making them unsuitable for real-time forecasting.
- Dependence on precise input data: Performance drops sharply if weather data is noisy, incomplete, or inaccurate.
- Poor generalisation: Models trained in one region struggle to adapt to different climatic conditions.
- Limited use of solar geometry: Many systems ignore azimuth, zenith, and incidence angles, reducing prediction accuracy.
- Instability in ML models: Without proper tuning and feature engineering, ML approaches may overfit and show inconsistent performance.

1.7 Input Design and Output Design

1.7.1 Input design

The input design acts as the primary interface between the user and the solar energy forecasting system, ensuring that raw solar and meteorological data are collected, validated, and transformed into a structured format suitable for processing **[1]**. The Data Input Module is responsible for accepting

various dataset formats, such as CSV, Excel, and real-time weather data streams, and converting them into a structured dataset for further analysis. This process involves preprocessing steps such as data cleaning, missing value handling, normalization, feature extraction, and format standardization to ensure compatibility with machine learning models such as Random Forest, Support Vector Regression, and Gradient Boosting Regressor.

In addition to handling solar and meteorological data, the system also incorporates user interaction features such as file selection, dataset validation, and system prompts to enhance usability. Validation mechanisms verify dataset format, parameter completeness, and data quality before processing begins, ensuring that only suitable inputs are accepted. Error-handling mechanisms such as warning messages and status indicators help users correct input mistakes and prevent system failures. Furthermore, the system ensures secure handling of data and maintains integrity throughout the processing pipeline. Overall, the input design emphasizes accuracy, consistency, usability, and reliability, forming a strong foundation for efficient solar energy forecasting and smart grid integration **[2].**

Objectives:

1. Converting Raw Solar Data into a Structured Format

The system transforms solar photovoltaic (PV) and meteorological data into a usable format suitable for machine learning models. It ensures compatibility through proper preprocessing and normalization while maintaining clean and consistent input data. Therefore, the system organizes these datasets into a standardized structure that can be easily interpreted by machine learning algorithms. This structured format enables efficient data processing, improves model learning, and ensures that the forecasting system can accurately analyze the relationship between weather conditions and solar power generation.

2. Creating User-Friendly Data Input Interface

The system provides a simple and interactive interface for uploading solar datasets. It supports CSV and Excel file selection and enables smooth and easy data handling for users with minimal technical knowledge. Users can easily browse and select files from their local systems without requiring complex configurations. This design improves usability and accessibility, allowing

researchers, engineers, and energy analysts to interact with the system efficiently while minimizing operational complexity.

3. Ensuring Data Accuracy and Validity

The system validates the dataset format and checks for missing values to ensure data accuracy. It prevents incorrect or corrupted data from entering the system and provides validation prompts and alerts to guide users. During the validation process, the system examines whether all required parameters such as solar irradiance, temperature, humidity, and wind speed are present in the dataset. If any inconsistencies or missing entries are detected, the system notifies the user and suggests corrective actions. This validation process plays an important role in maintaining high-quality data, which directly influences the reliability and accuracy of solar energy forecasting models.

4. Reducing Processing Errors and Time

The system cleans and preprocesses solar data to remove noise and redundancy. It improves the efficiency of model training and forecasting while avoiding system errors caused by improper input. Preprocessing techniques such as data filtering, normalization, and duplicate removal help streamline the dataset before it is used by machine learning algorithms. By eliminating irrelevant or inconsistent data entries, the system reduces computational overhead and speeds up the training process. This not only enhances prediction accuracy but also ensures that the forecasting system operates efficiently even when handling large solar datasets.

5. Supporting Efficient Data Preparation for Machine Learning Models

Another objective of the input design is to prepare the dataset in a way that is suitable for machine learning algorithms used in solar forecasting. The system extracts important features from solar and meteorological datasets and converts them into numerical values that can be processed by algorithms such as Random Forest, Support Vector Regression, and Gradient Boosting Regressor. Proper feature preparation helps the models identify meaningful relationships between environmental variables and solar power output, leading to improved prediction performance.

1.7.2 Output Design

The output design represents the final stage of the solar energy forecasting system, where processed data and prediction results are presented to the user in a clear and meaningful format **[1].** The primary goal of the output design is to display accurate solar photovoltaic (PV) power forecasts, performance metrics, and visual analysis in a user-friendly manner. The system generates prediction results using machine learning models such as Random Forest, Support Vector Regression, Decision Tree Regression, and Gradient Boosting Regressor, and presents them through graphs, tables, and comparative performance charts. These outputs help users understand solar power generation trends and evaluate the effectiveness of different forecasting models.

The system includes a visualization module that converts prediction results into graphical representations such as forecasting graphs, error comparison charts, and performance metric plots. Metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R^2 score are displayed to indicate the accuracy and reliability of the models. This enables users to interpret solar energy patterns and make informed decisions related to energy management and grid stability.

In addition to graphical outputs, the system generates structured reports and prediction summaries that can be stored or exported for further analysis. It also provides system notifications, status indicators, and error messages to inform users about processing and results. Overall, the output design focuses on clarity, accuracy, and usability, delivering meaningful insights for efficient solar energy utilization and smart grid operations.

2. LITERATURE SURVEY

1. ***A. R. Vijay Babu et al., IEEE Access (2026) — ML for enhanced grid stability: Positions solar forecasting as a direct tool for grid stability.*** This review provided a broad comparison of ML models, datasets, and techniques used in photovoltaic power forecasting. While offering extensive theoretical insights, the work did not include experimental testing or validation, limiting its relevance for real-world forecasting scenarios.

2. ***Nasab et al. (2025) – "Load Frequency Control in Renewable-Integrated Smart Grids Using Coordinated EV and HEV Support".*** This study introduced a coordinated control strategy to regulate load frequency in smart grids integrating renewable energy and hybrid electric vehicles. The method improved frequency stability under fluctuating solar inputs, but challenges such as high system complexity, communication constraints, and difficulty in synchronising multiple distributed energy resources limited large-scale deployment.

3. ***Tziolis et al. (2025) – "Machine Learning-Based Short-Term Net Load Forecasting for Solar Microgrids".*** Tziolis and colleagues proposed a machine-learning model for short-term net load prediction in solar-powered microgrids. Their system enhanced microgrid scheduling and operational efficiency. However, the approach was designed for small-scale systems and lacked scalability, reducing its applicability to large interconnected grid networks.

4. ***E. Swarnalatha et al., (2024) — "AI-enabled Microgrid Energy Management using Machine Learning".*** This work focuses on integrating machine learning techniques into microgrid energy management systems to optimize energy distribution from renewable sources such as solar wind. The study utilizes predictive models to forecast energy generation and consumption, enabling efficient load balancing and scheduling

5. ***Rajesh Tiwari et al., (2024) — "Neural Network-Based Forecasting for PV-Wind Microgrid".*** This study presents a neural network-based time series forecasting model to enhance power quality and stability in PV-wind hybrid microgrids. The model effectively captures complex patterns in solar and wind data, leading to improved prediction accuracy.

6. ***S. Radha Krishna Reddy et al., (2023) — "ANN-Based Voltage Control for Grid Stability".*** This research introduces an artificial neural network-based voltage control mechanism for maintaining grid stability under varying load and generation conditions. The model adapts dynamically to fluctuations in renewable energy inputs and ensures stable voltage levels.

7. ***Mohamed et al. (2022) – "Feature Engineering for Enhanced Solar Microgrid Forecasting".*** This study emphasised the role of feature engineering in improving solar forecasting accuracy. Their carefully derived features significantly enhanced model performance. Still, the approach was highly dependent on accurate weather data and was not validated across diverse climatic zones, limiting generalisation.

8. ***Piliougine et al. (2022) – "Machine Learning-Based Detection of Shading and Mismatch Losses in PV Arrays".*** The authors used machine-learning techniques to detect shading issues, mismatch losses, and PV array degradation. While effective for fault detection and maintenance, their work focused on diagnostics rather than forecasting solar power output.

9. ***Bouraiou et al. (2022) – "Performance and Failure Analysis of PV Pumping Systems in Saharan Climate Conditions".*** This field study examined PV pump systems operating in extreme Saharan environments. The authors identified key failure modes, including dust accumulation and overheating. Although valuable for reliability improvement, the work offered little contribution toward predictive forecasting.

10. ***Alam et al. (2021) – "Comparative Analysis of Traditional and Machine Learning Models for Solar Energy Forecasting".*** Alam compared statistical forecasting models with machine-learning techniques, demonstrating that ML outperformed classical methods in accuracy. However, the study used a limited dataset and lacked real-time testing, reducing practical applicability.

Focused Area / Title	Key Findings	Reference
ML for Enhanced Grid Stability [1]	Provides a broad comparison of machine learning models and techniques for photovoltaic power forecasting and highlights solar forecasting as a tool for improving grid stability, but lacks experimental validation for real-world deployment.	A. R. Vijay Babu et al., "ML for Enhanced Grid Stability," IEEE Access, 2026.
Load Frequency Control in Renewable Smart Grids [2]	Introduces coordinated EV and HEV support to regulate load frequency in renewable-integrated smart grids, improving stability under fluctuating solar inputs but increasing system complexity and communication requirements.	Nasab et al., "Load Frequency Control in Renewable-Integrated Smart Grids Using Coordinated EV and HEV Support," 2025ui.
Short-Term Net Load Forecasting for Solar Microgrids [3]	Proposes a machine learning model for short-term solar microgrid forecasting that improves scheduling and operational efficiency but is limited to small-scale systems and lacks scalability.	Tziolis et al., "Machine Learning-Based Short-Term Net Load Forecasting for Solar Microgrids," 2025.
AI-enabled Microgrid Energy Management [4]	Proposes an ML-based microgrid energy management system improving load balancing and efficiency, but lacks real-time grid stability analysis.	E. Swarnalatha et al., "AI enabled Microgrid Energy Management Systems using Machine Learning Algorithms: A Survey," 2024..
Neural Network-Based Forecasting for PV-Wind Microgrid [5]	Uses neural network-based forecasting to improve power quality and grid stability, enhancing	R. Tiwari et al., "Enhanced Power Quality and Forecasting for PV-Wind Microgrid Using Neural

	prediction accuracy and reducing voltage fluctuations.	Network-Based Time Series Forecasting," 2024.
ANN-Based Voltage Control for Grid Stability [6]	Uses an ANN-based voltage control system to improve grid stability and voltage regulation, but lacks forecasting for proactive management.	S. Radha Krishna Reddy et al., "Voltage Control of Self Excited Induction Generator using Artificial Neural Network," 2023.
Feature Engineering for Solar Forecasting [7]	Emphasizes feature engineering to improve solar forecasting accuracy and model performance, but depends heavily on accurate weather data and lacks climate diversity validation.	Mohamed et al., "Feature Engineering for Enhanced Solar Microgrid Forecasting," 2022.
PV Shading and Mismatch Loss Detection [8]	Uses machine learning to detect shading and mismatch losses in PV arrays, improving maintenance anddiagnostics but not focusing on solar forecasting.	Piliougine et al., "Machine Learning-Based Detection of Shading and Mismatch Losses in PV Arrays," 2022.
PV Pump System Performance in Extreme Climate [9]	Analyzes PV pumping systems in Saharan climate and identifies failure modes like dust and overheating, but does not address forecasting or grid stability.	Bouraiou et al., "Performance and Failure Analysis of PV Pumping Systems in Saharan Climate Conditions," 2022.
Traditional vs ML Solar Forecasting Models [10]	Compares traditional statistical models with machine learning techniques and shows ML improves accuracy, but uses limited datasets and lacks real-time validation.	Alam et al., "Comparative Analysis of Traditional and Machine Learning Models for Solar Energy Forecasting," 2021.

Table no.2.1. Literature Review Summary Table

3. SOFTWARE REQUIREMENTS ANALYSIS

3.1 Modules and Their Functionalities

3.1.1 Data Analysis

Data analysis plays a crucial role in understanding the nature and structure of the meteorological and solar photovoltaic (PV) datasets used for solar energy forecasting and grid stability analysis. The datasets consist of historical solar power generation data collected under varying environmental and climatic conditions, including fluctuations in solar irradiance, temperature, humidity, wind speed, cloud cover, and seasonal variations. These datasets often include complex scenarios such as sudden weather changes, irregular solar radiation patterns, and varying power output across different time intervals. The presence of multiple environmental parameters and their interdependencies increases the complexity of the dataset. Additionally, challenges such as missing values, sensor noise, inconsistent data intervals, and extreme weather conditions require careful analysis during the data evaluation phase.

The dataset may contain noise and inconsistencies such as incomplete records, outliers, measurement errors, and fluctuations in solar power due to cloud shading or atmospheric disturbances. These factors can negatively impact the performance of machine learning models if not properly handled. Therefore, detailed data analysis is essential to identify such issues and design effective preprocessing and feature engineering strategies. This understanding helps in selecting appropriate data cleaning methods, normalization techniques, and model parameters for accurate forecasting.

3.1.2 Data Preprocessing

Data preprocessing converts raw solar photovoltaic (PV) and meteorological data into a clean and structured format suitable for machine learning models. This stage includes handling missing values, removing outliers, smoothing noisy sensor readings, and aligning timestamps across different data sources such as solar irradiance, temperature, humidity, wind speed, and historical PV output. Standardization and normalization techniques are applied to ensure consistent data

distribution, reduce data imbalance, and improve model convergence and forecasting accuracy. Proper preprocessing helps minimize errors and ensures that the dataset is reliable for further analysis and model training.

In addition, time-based features such as hour, day, month, and seasonal variations are extracted to capture daily and yearly solar generation patterns. Weather parameters are transformed using rolling averages, lag variables, and trend smoothing to better represent environmental conditions and their influence on solar power output. These feature engineering techniques help the model understand temporal dependencies and complex relationships between weather factors and PV generation. The cleaned and well-structured dataset forms a strong foundation for accurate solar energy forecasting, efficient grid stability management, and consistent performance under different climatic conditions.

3.1.3 Machine Learning Algorithm for Prediction

The proposed system utilizes an ensemble-based machine learning approach to improve solar energy prediction accuracy and stability. Two complementary models, Gradient Boosting Regressor (GBR) and Random Forest Regressor (RFR), are selected due to their strong ability to capture complex relationships between meteorological parameters and solar power generation. These models analyze features such as solar irradiance, temperature, humidity, wind speed, and historical PV output to learn both linear and non-linear patterns in the dataset.

Both models operate independently and generate separate prediction outputs, which are then combined using a weighted averaging or soft-voting mechanism to produce the final solar energy forecast. This ensemble approach enhances robustness against sudden weather changes, reduces overfitting, and improves prediction reliability. As a result, the system provides accurate short-term and medium-term solar energy forecasts, supporting better grid stability and efficient energy management.

3.2 Functional Requirements

The functional requirements define the essential operations that the solar energy forecasting system must perform to ensure accurate prediction and efficient performance. The system is designed to process meteorological and solar photovoltaic (PV) data, extract meaningful patterns, and generate reliable energy forecasts that support grid stability and energy management. It integrates multiple modules such as data input, preprocessing, machine learning prediction, and visualization to ensure a smooth and efficient workflow.

Additionally, the system focuses on providing accurate and timely solar energy forecasts to assist energy planners, grid operators, and researchers in decision-making. It is designed to be flexible and scalable, allowing the integration of advanced machine learning models, real-time data sources, and intelligent analytics in the future. The functional requirements also ensure that the output is visually clear and analytically useful, enabling users to easily interpret forecasting results and understand solar power generation trends. Overall, these requirements define how effectively the system performs its core functions and delivers reliable forecasting results.

transforms raw weather and solar data into actionable energy forecasting insights **[1]**.

1. The system shall accept meteorological and historical solar PV data as input for forecasting.
2. The system shall preprocess and clean the input data by handling missing values, noise, and inconsistencies.
3. The system shall apply ensemble machine learning models to generate accurate solar energy forecasts.
4. The system shall analyze weather parameters and predict short-term and medium-term solar power output.
5. The system shall generate graphical and numerical outputs such as charts, tables, and prediction reports for users.

3.3 Non-Functional Requirements

The non-functional requirements define the quality attributes and operational constraints of the solar energy forecasting system. These requirements focus on how efficiently and reliably the system performs under different environmental and operational conditions. The system must handle various types of meteorological and solar photovoltaic (PV) datasets, including variations in data size, time intervals, and sensor inputs, while maintaining consistent forecasting accuracy and stable performance. It should ensure smooth execution without failures or interruptions, even when processing large volumes of weather and solar data.

Furthermore, the system is expected to provide fast processing with minimal latency to support near real-time solar energy forecasting and grid stability monitoring. Scalability is an important factor, allowing the system to adapt to increasing data sources, additional sensors, and advanced

machine learning models in the future. Security and data integrity are also essential to ensure that meteorological data, solar datasets, and forecasting outputs are handled safely and confidentially. These requirements ensure that the system is robust, efficient, and suitable for real-world deployment in solar energy forecasting and smart grid applications.

The system shall ensure high reliability and stable performance during solar data processing and forecasting.

1. The system shall provide scalable performance across different data sizes, time intervals, and meteorological sources.
2. The system shall maintain efficient processing with minimal latency for near real-time solar energy prediction.
3. The system shall ensure secure handling of solar and weather datasets while maintaining confidentiality and data integrity.

3.4 Feasibility Study

The feasibility study evaluates whether the proposed solar forecasting system can be realistically developed based on technical, economic, and social considerations. The availability of open-source forecasting tools, meteorological datasets, and computational resources supports the development process. The proposed architecture is cost-effective, scalable, and suitable for integration with real-world energy management platforms.

Key Considerations in Feasibility Analysis :

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

3.4.1 Economic Feasibility

The system relies on open-source libraries (Python, Scikit-learn), free meteorological datasets, and standard computing infrastructure, keeping implementation costs low. No specialized hardware or paid tools are required, making the solution economically feasible for academic, research, and industrial use.

3.4.2 Technical Feasibility

The required machine learning techniques, preprocessing tools, and forecasting algorithms are well-established and supported by widely used platforms. Tools such as Python, Pandas, NumPy, and Scikit-learn provide strong technical support for implementation. The system is compatible with common computing resources, making technical deployment practical and achievable.

3.4.3 Social Feasibility

Accurate solar forecasting contributes to sustainable energy use and supports society's shift toward renewable energy. Power companies, consumers, and policy-makers benefit from improved grid stability and reduced energy wastage. Since renewable energy management is a global priority, the system is expected to receive strong social acceptance and support.

4. SYSTEM SPECIFICATIONS

4.1 Software Requirements

The software requirements define the essential tools, libraries, and platforms necessary to design, implement, and deploy an effective solar energy forecasting system. This system involves multiple tasks, including data collection, preprocessing, feature engineering, model training, prediction generation, and result visualization. A robust software environment ensures seamless integration of these modules, maintains stability during computation-intensive tasks, and allows scalability for future enhancements such as real-time forecasting or integration with smart grid systems. Proper software planning also reduces development complexity, improves reproducibility, and ensures compatibility across different hardware and operating systems.

To process large meteorological and PV datasets efficiently, the system relies on high-performance programming libraries. Python 3.x is chosen as the primary programming language due to its versatility, extensive support for scientific computing, and compatibility with machine learning frameworks. Libraries such as Pandas and NumPy are used for data manipulation and numerical computations, enabling efficient handling of large datasets. Scikit-learn supports building and evaluating machine learning models, while XGBoost and LightGBM provide gradient boosting implementations for accurate solar energy predictions. For visualization, Matplotlib and Seaborn are employed to generate clear graphs and plots, assisting in the analysis of solar patterns and forecasting results.

Operating System: Windows

Programming Language: Python 3.x

Core Libraries/Frameworks: Pandas, NumPy, Scikit-learn, Matplotlib, Seaborn, XGBoost, LightGBM

Development Environment: VS Code / PyCharm / Jupyter Notebook

Database Tools: MySQL (WAMP Server)

Documentation Tools: MS Word

4.2 Hardware Requirements

The hardware requirements define the minimum computational resources necessary for developing and running the proposed Solar Energy Forecasting System. Since the project involves processing large meteorological and PV datasets, performing data preprocessing, training ensemble machine learning models, and generating solar energy predictions, the system requires a computer with sufficient processing capability, memory, and storage. During the initial stages of development, testing, and experimentation, standard computing hardware such as a laptop or desktop with moderate specifications is generally sufficient. This allows the system to perform basic operations such as data cleaning, feature engineering, and model evaluation without the need for highly advanced equipment.

However, for achieving faster execution, especially when working with large datasets or training machine learning models like Gradient Boosting Regressor (GBR) and Random Forest Regressor (RFR), GPU support becomes highly beneficial. A dedicated graphics processing unit can significantly accelerate model training, reduce computation time, and improve prediction efficiency. Adequate RAM and storage are also important for handling datasets, saving trained models, and storing processed outputs efficiently. The recommended hardware configuration ensures that the system remains stable, scalable, and compatible with future requirements such as advanced model tuning, extended training, and deployment in real-world solar forecasting environments.

Processor: Intel Core i3/i5 or equivalent (Pentium IV minimum)

Memory (RAM): Minimum 8 GB

Storage: 512 GB HDD/SSD

Display/Monitor: SVGA or higher

Input Devices: Standard keyboard and two/three-button mouse

5. SOFTWARE DESIGN

5.1 System Architecture

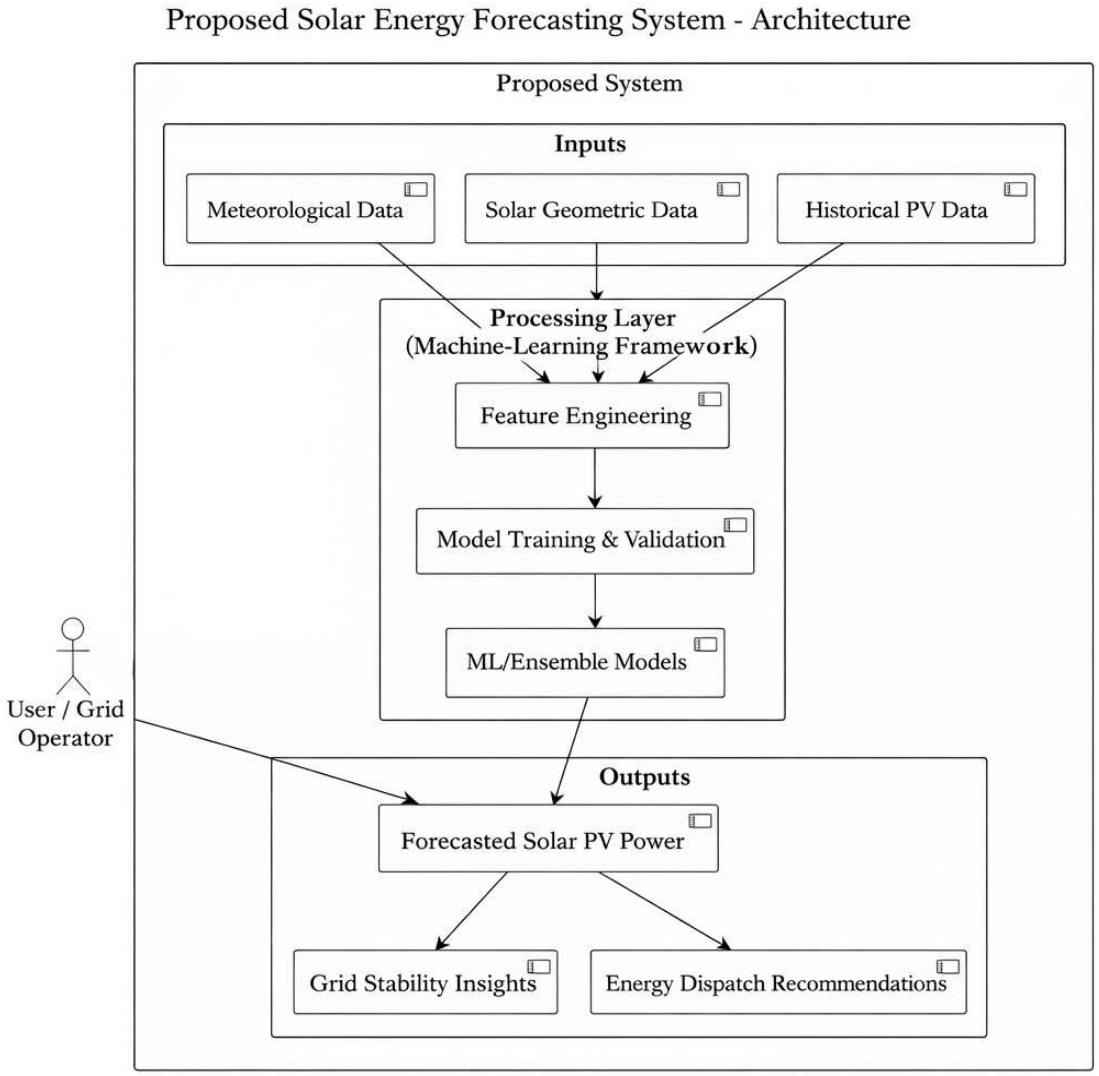

Figure 5.1 System Architecture

The proposed solar energy forecasting system is designed to provide accurate and reliable predictions of photovoltaic (PV) power generation by integrating multiple data sources **[1], [2], [4]**. These sources include meteorological data such as temperature, solar irradiance, humidity, wind speed, and cloud cover, along with solar geometric data like sun angles, solar altitude, and azimuth,

as well as historical PV generation records . All input data undergoes preprocessing to remove noise, normalize values, and extract relevant features that significantly impact solar energy output **[7], [8]**.

The processed features are then used to train advanced machine learning models. Ensemble-based approaches, combining multiple algorithms, are employed to improve forecasting accuracy, reduce prediction variance, and enhance model robustness**.** Model validation ensures that predictions generalise well to unseen data, while probabilistic outputs allow for more reliable decision-making **[5], [11]**.

The system generates forecasted solar power output, which can be used by grid operators for effective energy planning, maintaining voltage and frequency stability, optimising energy dispatch, and managing storage utilisation **.** The modular and scalable architecture supports real-time deployment and adaptation to diverse datasets **.** By leveraging historical trends, real-time data, and ensemble learning, this system facilitates informed decision-making, supports renewable energy integration, and enhances long-term grid reliability and sustainability **[[10]**.

5.2 Dataflow Diagram

A **Data Flow Diagram (DFD)** is a structured analysis and design tool that graphically represents the movement of data within a system. For the Solar Energy Forecasting System, the DFD illustrates how data is collected from multiple sources, processed through various stages, stored for future use, and ultimately used to generate accurate solar power predictions. It provides a high-level view of the system's functionality while maintaining clarity and modularity.

External Entities: These are sources or destinations of data outside the system. Examples include weather stations providing meteorological data, solar panels generating real-time PV output, and grid operators requiring forecast information for operational planning. These entities interact with the system by supplying input data or receiving processed outputs.

Processes: Processes are operations or transformations applied to incoming data. In this system, processes include data preprocessing (handling missing values, smoothing, and normalization), feature extraction (deriving time-based and weather-based features), machine learning model training, ensemble prediction generation, and forecast delivery. Each process ensures that raw data is transformed into accurate, actionable information.

Data Stores: Data stores serve as repositories for all system data. This includes historical solar energy records, meteorological datasets, preprocessed data, and trained machine learning models. Properly maintained data stores allow the system to reference past data for improving prediction accuracy and support model retraining or updates.

Data Flows represent the movement of information between external entities, processes, and data stores in the system. They illustrate how data travels from input collection through processing and storage to the final output. Data Flow Diagrams (DFDs) help designers and stakeholders understand system operations without focusing on code. Level 0 shows the system as a single process interacting with external entities, Level 1 breaks it into major modules such as data acquisition, preprocessing, model training, and forecasting, and Level 2 further details internal steps like handling missing data, feature engineering, and prediction generation.

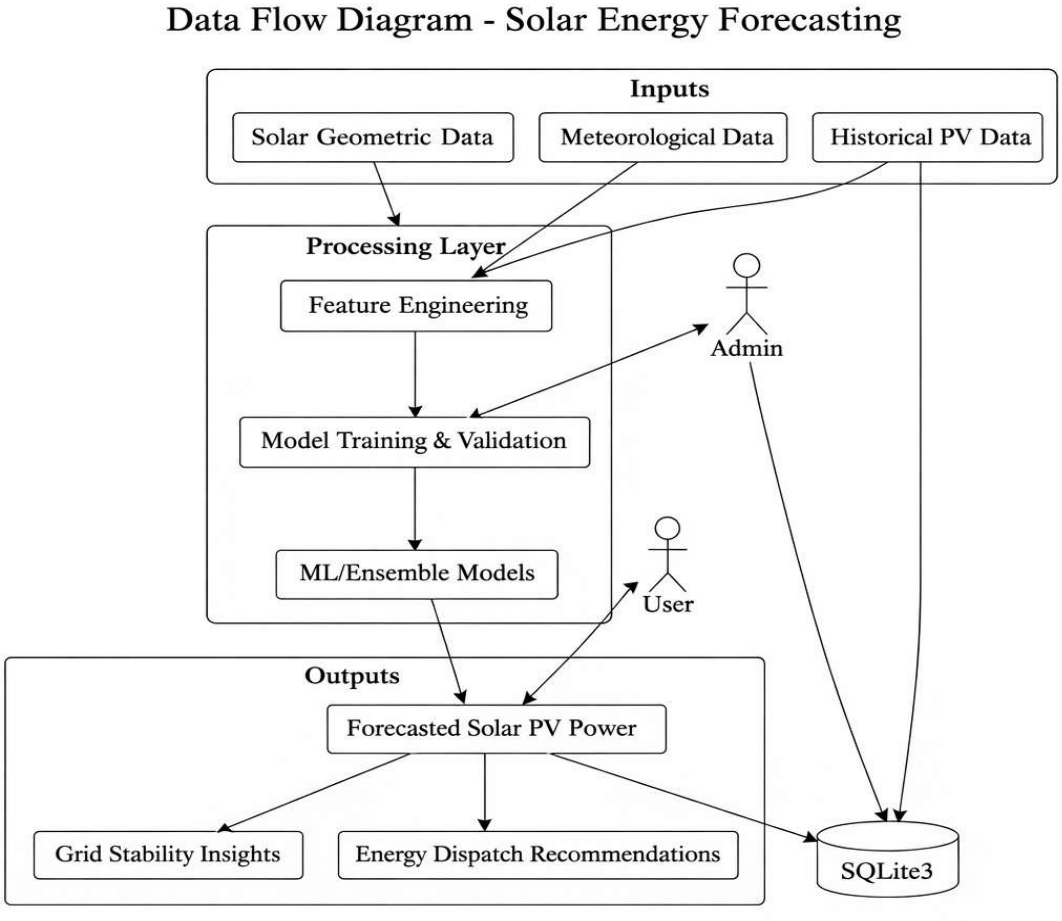

Figure 5.2 Dataflow Diagram

5.3 UML Diagrams

UML (Unified Modelling Language) is a standardised visual modelling language used for specifying, visualising, constructing, and documenting the components and behaviour of software systems. It provides a clear and systematic way to represent how a system is designed and how its different parts interact with one another.

It was developed to simplify the process of software design by offering a common language that can be understood by developers, analysts, designers, and other stakeholders involved in the system development process. It plays an important role in software engineering because it helps transform abstract system ideas—such as solar energy forecasting, data preprocessing, and predictive modelling—into structured diagrams that are easier to interpret, communicate, and implement. Created and standardised by the Object Management Group (OMG), UML 1.0 was formally proposed in January 1997, and since then, it has become one of the most widely used modelling tools in software design.

UML is closely associated with object-oriented analysis and design, making it especially useful for systems that are developed using modular and reusable software principles. It allows developers to describe both the structure and behaviour of a system before actual coding begins, thereby reducing design errors and improving software quality.

These diagrams help in identifying the key elements of a system, their attributes, methods, relationships, and the way external entities interact with the system. In the case of a Solar Energy Forecasting System, UML can be used to represent the interactions between external entities (such as weather stations, solar panels, and grid operators), input processing modules, data preprocessing and feature extraction engines, ensemble-based prediction models, and forecast output modules. By visualising these interactions and internal structures, developers can better understand system flow, dependencies, and implementation requirements.

UML diagrams are generally divided into two major categories: Behavioural Diagrams and Structural Diagrams. Behavioural UML diagrams focus on the dynamic aspects of the system, describing how the system behaves over time, how external entities interact with it, and how different modules communicate during execution. Examples include Use Case Diagrams, Activity

Diagrams, and Sequence Diagrams. In contrast, Structural UML diagrams represent the static architecture of the system by showing its classes, objects, components, and their relationships. Examples include Class Diagrams, Component Diagrams, and Deployment Diagrams. UML has been widely accepted as an industry standard because it provides a formal basis for understanding software design, supports effective communication among team members, and encourages the use of best practices in system development.

Goals of UML for Solar Energy Forecasting System:

- To provide an expressive and standardised visual modelling language for designing the forecasting system.
- To help developers represent system structure, behaviour, and interactions clearly.
- To establish a formal basis for understanding and documenting solar energy forecasting models.
- To simplify communication between developers, energy analysts, and stakeholders.
- To support object-oriented analysis and design principles effectively.
- To reduce system design complexity by breaking the forecasting system into manageable components.
- To assist in identifying system requirements, modules, and relationships before coding begins.
- To improve software quality by enabling better planning and error detection during design.
- To encourage the growth and use of object-oriented tools and CASE tools.
- To provide reusable and scalable design models for future enhancements, such as integrating additional weather sensors or advanced ML models.

Types of UML Diagrams:

1 Use Case Diagram:
2 Class Diagram:
3 Sequence Diagram:
4 Activity Diagram:

5.3.1 Use Case Diagram

The Use Case Diagram represents the interaction between external actors and the Solar Energy Forecasting System **.** It provides a high-level view of how different users and system components participate in the forecasting process . In this diagram, the main actors include the User and Admin, each with distinct roles in interacting with the system **.**The use case diagram helps in identifying the major functionalities provided by the system and illustrates how these functionalities are connected to the actors .

The process begins when the User interacts with the system by initiating forecast requests, viewing energy dispatch recommendations, or checking grid insights . These functionalities enable the user to gain actionable information on solar energy generation, grid stability, and energy management strategies **.**

The Admin actor is responsible for monitoring system performance, updating prediction models, and accessing the database. These functions ensure that the forecasting system operates efficiently, maintains accuracy, and can be updated with new data or improved models

Components of the Use Case Diagram:

User:

The User interacts with the system to request forecasts, view grid insights, and access energy dispatch recommendations for decision-making.

Admin:

The Admin manages system monitoring, updates forecasting models, and accesses the database to ensure smooth operation and maintain data integrity.

Request Forecast:

This use case allows the user to input a forecast request for a specific time range or scenario, initiating the prediction process.

View Grid Insights:

The user can visualize grid-related insights, such as load balancing, energy availability, and predicted generation trends.

View Energy Dispatch Recommendations:

The system provides actionable recommendations for energy dispatch and storage management based on forecasted solar output.

Monitor System:

The admin monitors system performance, ensuring models are operating correctly and forecasts are reliable.

Update Models:

The admin updates and fine-tunes machine learning models for improved forecasting accuracy.

Access Database (SQLite3):

The admin can access stored historical data, meteorological records, and model outputs.

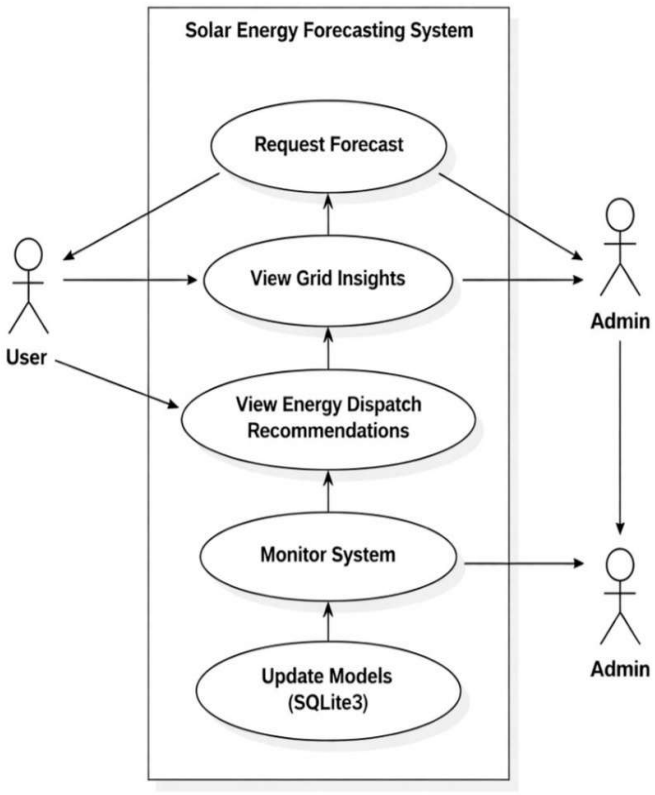

Figure 5.3.1 Use Case Diagram

5.3.2. Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

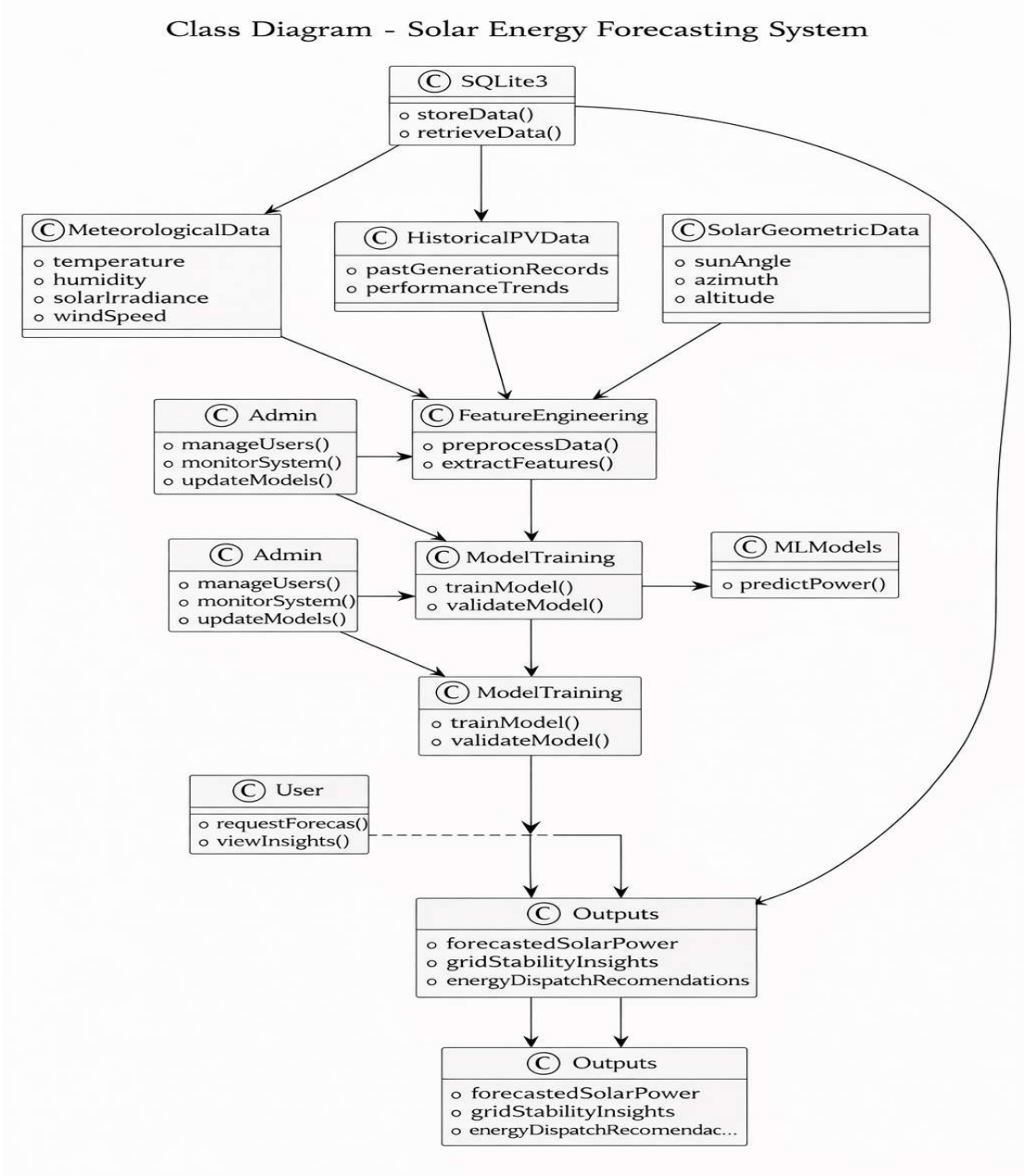

Figure 5.3.2 Class Diagram

1. **SQLite3**

The SQLite3 component represents the database layer of the system, responsible for storing, managing, and retrieving all relevant data used in the forecasting process. It maintains structured datasets including meteorological data, solar geometric parameters, historical photovoltaic (PV) generation records, and forecasted outputs. The methods storeData() and retrieveData() facilitate efficient data persistence and access. This component ensures data consistency, supports model training, and enables future analysis, thereby playing a crucial role in maintaining system reliability.

2. **MeteorologicalData**

The MeteorologicalData component is responsible for handling weather-related parameters that significantly influence solar energy generation. It includes attributes such as temperature, humidity, solar irradiance, and wind speed. These environmental variables are critical inputs to the forecasting models, as they directly impact the availability and intensity of solar energy. Accurate representation of meteorological conditions enhances the effectiveness of the prediction process.

3. **SolarGeometricData**

The SolarGeometricData component captures solar position-related parameters such as sun angle, azimuth, and altitude. These attributes describe the orientation and intensity of sunlight incident on solar panels at a given time. Incorporating this information enables the system to better understand solar exposure patterns, thereby improving the precision of energy forecasting models.

4. **HistoricalPVData**

The **HistoricalPVData** component stores past records of solar photovoltaic (PV) power generation along with performance trends over time. This time-series dataset includes historical measurements such as solar irradiance, temperature, humidity, and corresponding PV output collected from solar monitoring systems. The stored historical data plays a critical role in training machine learning models, as it enables the system to analyze long-term patterns, seasonal variations, and temporal dependencies that influence solar energy production.

5. FeatureEngineering

The FeatureEngineering component is responsible for preprocessing and transforming raw input data into a structured and meaningful format suitable for machine learning. It performs operations such as data cleaning, handling missing values, normalization, and feature extraction through the methods preprocessData() and extractFeatures(). This stage ensures that only relevant and high-quality features are used, thereby improving model accuracy and efficiency.

6. ModelTraining

The ModelTraining component manages the development and validation of machine learning models. It utilizes processed data from the feature engineering stage to train predictive models and evaluate their performance. The methods trainModel() and validateModel() ensure that the models are reliable, accurate, and capable of generalizing to new data. This component is critical for building a robust forecasting system.

7. MLModels

The MLModels component represents the execution layer where trained machine learning models are deployed for prediction. It is responsible for generating solar PV power forecasts using the method predictPower(). By applying learned patterns to new input data, this component produces accurate predictions that support decision-making in energy management.

8. User

The **User** component represents the end user who interacts with the system to request forecasts and access analytical insights. It provides methods such as **requestForecast()** to initiate prediction processes and **viewInsights()** to visualize results. This component ensures that the system remains user-friendly and accessible to stakeholders such as engineers, analysts, and grid operators. In addition, the user can upload solar photovoltaic and meteorological datasets, configure forecasting parameters, and monitor prediction results through an interactive interface. The system provides visual outputs such as charts, graphs, and statistical summaries that help users understand predicted solar power generation trends and compare them with historical data.

9. Admin

The Admin component represents the system administrator responsible for overseeing system operations and maintenance. It includes functionalities such as manageUsers(), monitorSystem(), and updateModels(). The admin ensures smooth system performance, manages user access, and updates machine learning models to maintain forecasting accuracy under changing conditions.

10. Outputs

The Outputs component represents the final results generated by the system. It includes forecasted solar PV power, grid stability insights, and energy dispatch recommendations. These outputs provide actionable information that supports efficient energy management and decision-making. This component acts as the final stage of the system, delivering meaningful results to users and stakeholders.

5.3.3. Sequence Diagram

The Sequence Diagram represents the dynamic interactions between different modules of the Solar Energy Forecasting System in a time-ordered manner . It illustrates how the user initiates the system by submitting a forecast request and how the internal components collaborate step by step to process input data and generate accurate solar energy predictions . Unlike structural diagrams that focus on system organization, the sequence diagram emphasizes the flow of messages and operations exchanged between modules during execution . This approach helps in understanding the chronological order of system behavior and clarifies how each module contributes to the overall forecasting process, ensuring reliable and timely solar energy forecasts.

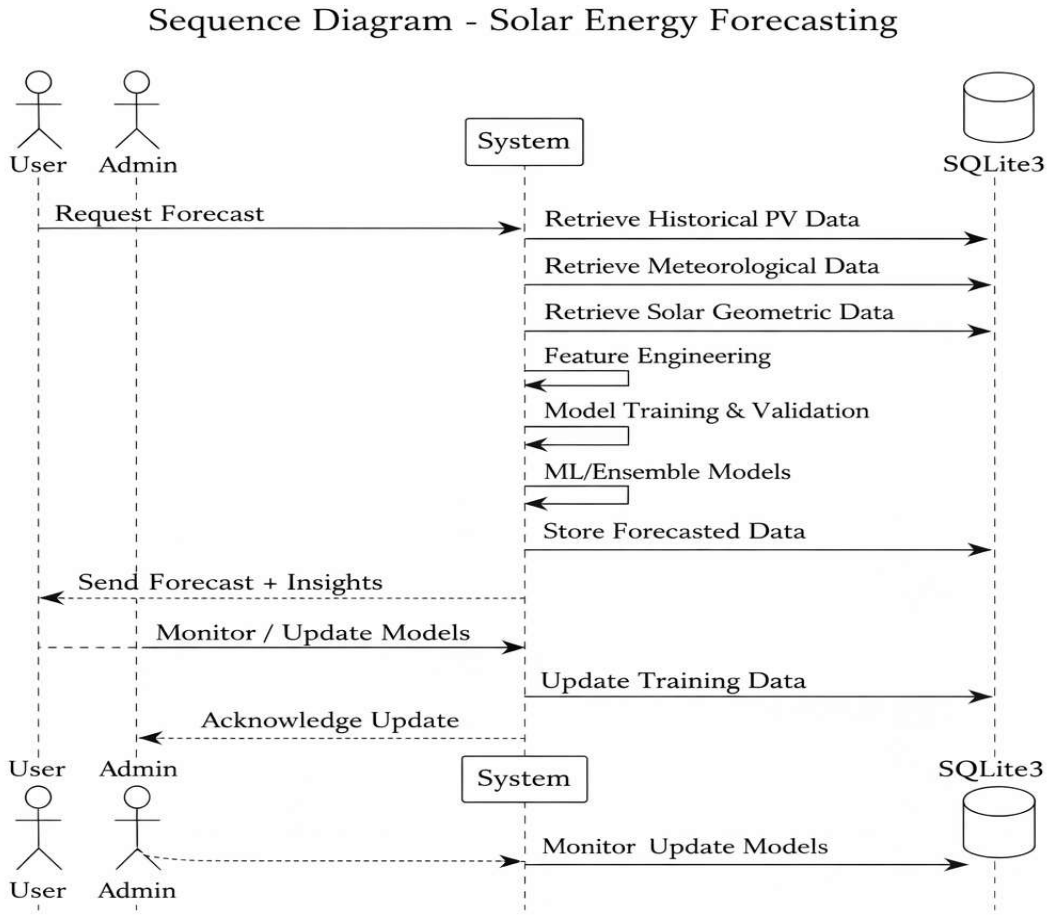

Figure 5.3.3 Sequence Diagram

The sequence begins with the User, who submits a forecast request by specifying parameters such as location, date/time range, and forecast horizon. The request is received by the InputProcessor

module, which validates the request and retrieves relevant data from the Database. The database supplies historical photovoltaic (PV) generation data, meteorological data (temperature, solar irradiance, humidity, wind speed, cloud cover), and solar geometry parameters.

Next, the PreprocessingModule cleans and prepares the data by handling missing values, removing outliers, and normalising the features. Feature engineering is applied to generate time-based and derived variables such as rolling averages, lag variables, and seasonal indicators. The processed dataset is then sent to the ForecastingModel module, where ensemble machine learning techniques, such as a combination of Gradient Boosting Regressor (GBR) and Random Forest Regressor (RFR), are applied to produce robust predictions. The ensemble setup ensures model stability and accuracy, even under varying weather conditions.

Once the forecasts are generated, the EvaluationModule computes performance metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R^2 score. Visualisation components generate graphs, charts, and tables to present forecast results in a clear and interpretable manner. Finally, the OutputHandler delivers the forecasted solar energy data along with actionable recommendations for energy management, grid stability, and storage optimisation. This sequence diagram clearly illustrates the logical communication flow of the system and demonstrates how raw historical and meteorological data are transformed into reliable solar energy forecasts.

List of actions:

- **User:** Initiates the process by submitting a forecast request with specific parameters. Receives the forecast results in graphical, tabular, or numerical form for decision-making.
- **System:** Validates the forecast request, retrieves data from the database, and ensures data is clean, consistent, and aligned for accurate processing.
- **Model:** Applies preprocessing, feature engineering, and ensemble machine learning algorithms to generate accurate and reliable solar energy predictions.
- **Evaluation:** Computes performance metrics, validates predictions, generates visual insights, and delivers actionable recommendations to the user.

5.3.4 Activity Diagram

Activity diagrams are graphical representations of work flows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step work flows of components in a system. An activity diagram shows the overall flow of control.

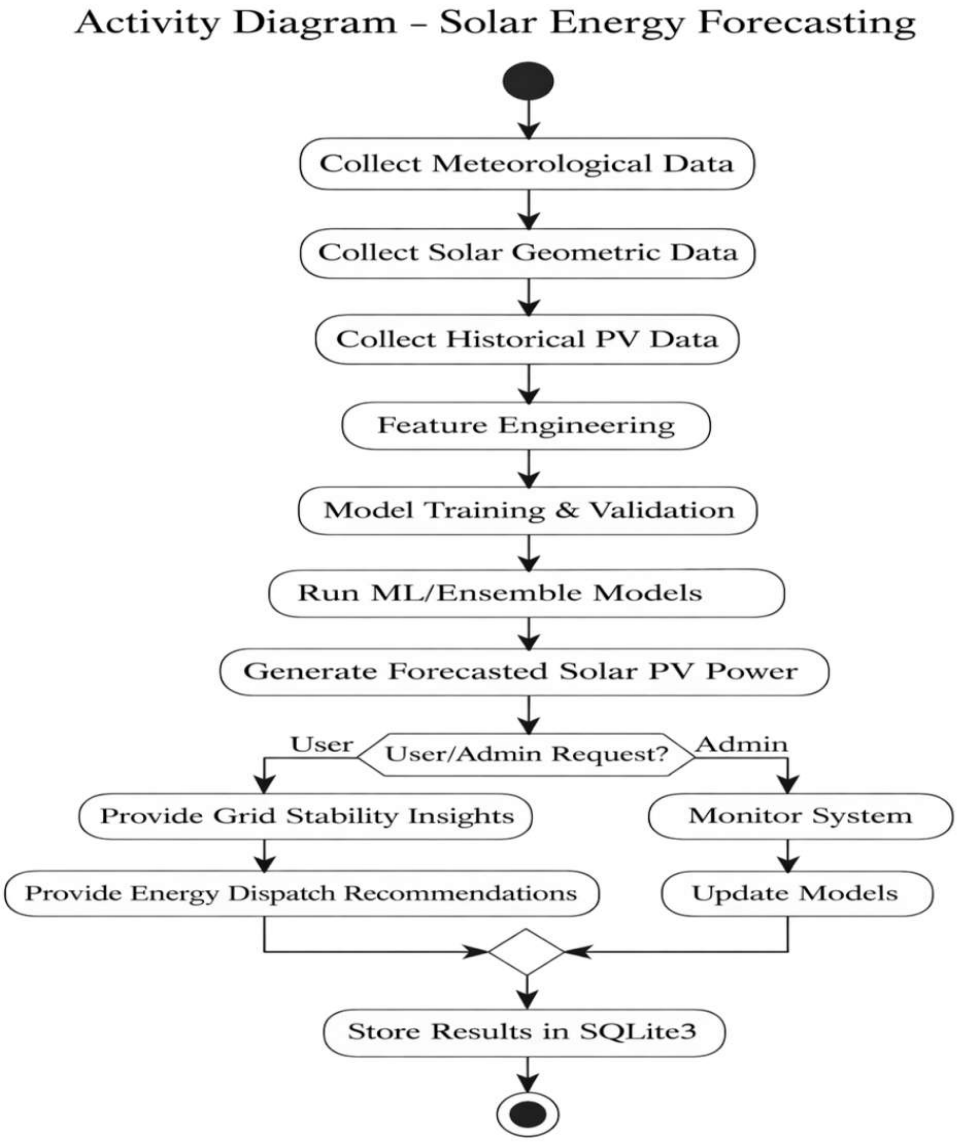

Figure 5.3.4 Activity Diagram

Activity Diagram Components

1. Start Node

The Start Node denotes the initiation of the system workflow. It represents the initial state from which all data processing and analytical operations are triggered.

2. Collect Meteorological Data

This activity involves the acquisition of weather-related parameters, including temperature, humidity, wind speed, and cloud cover. These variables are critical determinants of solar energy generation and serve as primary inputs to the forecasting model.

3. Collect Solar Geometric Data

This stage captures solar-specific parameters such as solar irradiance, zenith angle, and azimuth angle. These parameters define the intensity and orientation of sunlight incident on solar panels, thereby influencing energy output.

4. Collect Historical PV Data

This activity focuses on gathering historical records of solar power generation. The data provides temporal insights and enables the machine learning models to learn underlying patterns and trends in energy production.

5. Feature Engineering

Feature engineering is performed to preprocess and transform raw data into a structured format suitable for model training. It includes data cleaning, normalization, feature selection, and the generation of derived attributes to enhance predictive accuracy.

6. Model Training & Validation

In this stage, machine learning models are trained using the processed dataset. The models are subsequently validated using appropriate evaluation techniques to ensure their reliability, accuracy, and generalization capability.

7. Run ML / Ensemble Models

This activity involves the execution of trained models to generate predictions. Ensemble methods may be employed to combine multiple models, thereby improving prediction robustness and reducing variance.

8. Generate Forecasted Solar PV Power

The system produces forecasts of solar PV power output for future time intervals. These predictions are essential for planning and managing power system operations.

Forecasted solar PV power generation refers to the prediction of future electricity output from solar panels based on historical generation data and environmental factors such as solar irradiance, temperature, and weather conditions. Depending on the time horizon, forecasting can be short-term (minutes to hours), medium-term (hours to days), or long-term (weeks to months), each serving different operational and planning purposes in power systems.

9. Decision Node – User/Admin Request

This decision node determines the type of request received by the system. Based on the classification, the workflow is directed toward either user-oriented outputs or administrative functions.

10. Provide Grid Stability Insights (User Path)

For user-level requests, the system analyzes forecasted solar output and provides insights into grid stability, including potential fluctuations, risks, and operational impacts.

11. Provide Energy Dispatch Recommendations (User Path)

The system generates optimized energy dispatch strategies aimed at maintaining equilibrium between energy supply and demand, thereby enhancing overall grid efficiency.

12. Monitor System (Admin Path)

This activity enables administrators to monitor system performance, track forecasting accuracy, and ensure smooth operation of the forecasting pipeline.

13. Update Models (Admin Path)

Administrators can update or retrain machine learning models using new data to maintain adaptability and accuracy under evolving environmental conditions.

14. Store Results in SQLite3

All outputs, including predictions, analytical insights, and logs, are stored in a SQLite3 database to ensure data persistence, traceability, and future accessibility.

15. End Node

The End Node signifies the completion of the workflow. It confirms that all stages—from data collection to prediction and storage—have been successfully execute

6. CODING AND IMPLEMENTATION

6.1 Source Code

Frontend

css

```
select.admin-autocomplete {
    width: 20em;
}
.select2-container--admin-autocomplete.select2-container {
    min-height: 30px;
}
.select2-container--admin-autocomplete .select2-selection--single,
.select2-container--admin-autocomplete .select2-selection--multiple {
    min-height: 30px;
    padding: 0;
}
.select2-container--admin-autocomplete.select2-container--focus .select2-selection,
.select2-container--admin-autocomplete.select2-container--open .select2-selection {
    border-color: var(--body-quiet-color);
    min-height: 30px;
}
.select2-container--admin-autocomplete.select2-container--focus.select2-selection.select2-
selection--single,
.select2-container--admin-autocomplete.select2-container--open.select2-selection.select2-
selection--single {
    padding: 0;
}
.select2-container--admin-autocomplete.select2-container--focusselect2-selection.select2-
selection--multiple,
.select2-container--admin-autocomplete.select2-container--open.select2-selection.select2-
selection--multiple {
    padding: 0;
```

```
}
.select2-container--admin-autocomplete .select2-selection--single {
    background-color: var(--body-bg);
    border: 1px solid var(--border-color);
    border-radius: 4px;
}
.select2-container--admin-autocomplete .select2-selection--single .select2-selection__rendered {
    color: var(--body-fg);
    line-height: 30px;
}
.select2-container--admin-autocomplete .select2-selection--single .select2-selection__clear {
    cursor: pointer;
    float: right;
    font-weight: bold;
}
.select2-container--admin-autocomplete  .select2-selection--single  .select2-selection__placeholder
{
    color: var(--body-quiet-color);
}
.select2-container--admin-autocomplete .select2-selection--single .select2-selection__arrow {
    height: 26px;
    position: absolute;
    top: 1px;
    right: 1px;
    width: 20px;
}
.select2-container--admin-autocomplete .select2-selection--single .select2-selection__arrow b {
    border-color: #888 transparent transparent transparent;
    border-style: solid;
    border-width: 5px 4px 0 4px;
    height: 0;
    left: 50%;
    margin-left: -4px;
    margin-top: -2px;
```

```
    position: absolute;
    top: 50%;
    width: 0;
}
.select2-container--admin-autocomplete[dir="rtl"].select2-selection--singleselect2-
selection__clear {
    float: left;
}
.select2-container--admin-autocomplete[dir="rtl"].select2-selection--singleselect2-
selection__arrow {
    left: 1px;
    right: auto;
}
.select2-container--admin-autocomplete.select2-container--disabled .select2-selection--single {
    background-color: var(--darkened-bg);
    cursor: default;
}
.select2-container--admin-autocomplete.select2-container--disabledselect2-selection--single
.select2-selection__clear {
    display: none;
}
.select2-container--admin-autocomplete.select2-container--open .select2-selection--single .select2-
selection__arrow b {
    border-color: transparent transparent #888 transparent;
    border-width: 0 4px 5px 4px;
}
.select2-container--admin-autocomplete .select2-selection--multiple {
    background-color: var(--body-bg);
    border: 1px solid var(--border-color);
    border-radius: 4px;
    cursor: text;
}
.select2-container--admin-autocomplete .select2-selection--multiple .select2-selection__rendered {
    box-sizing: border-box;
```

```
    list-style: none;
    margin: 0;
    padding: 0 10px 5px 5px;
    width: 100%;
    display: flex;
    flex-wrap: wrap;
}

.select2-container--admin-autocomplete .select2-selection--multiple .select2-selection__rendered li
{
    list-style: none;
}
.select2-container--admin-autocomplete.select2-selection--multipleselect2-selection__placeholder
{
    color: var(--body-quiet-color);
    margin-top: 5px;
    float: left;
}
.select2-container--admin-autocomplete .select2-selection--multiple .select2-selection__clear {
    cursor: pointer;
    float: right;
    font-weight: bold;
    margin: 5px;
    position: absolute;
    right: 0;
}
.select2-container--admin-autocomplete .select2-selection--multiple .select2-selection__choice {
    background-color: var(--darkened-bg);
    border: 1px solid var(--border-color);
    border-radius: 4px;
    cursor: default;
    float: left;
    margin-right: 5px;
    margin-top: 5px;
```

```
    padding: 0 5px;
}

.select2-container--admin-autocomplete.select2-selection--multiple.select2-
selection__choice__remove {
    color: var(--body-quiet-color);
    cursor: pointer;
    display: inline-block;
    font-weight: bold;
    margin-right: 2px;
}
.select2-container--admin-autocompleteselect2-selection--multipleselect2-
selection__choice__remove:hover {
    color: var(--body-fg);
}
.select2-container--admin-autocomplete[dir="rtl"].select2-selection--multiple.select2-
selection__choice,     .select2-container--admin-autocomplete[dir="rtl"].select2-selection--multiple
.select2-selection__placeholder,select2-container--admin-autocomplete[dir="rtl"].select2-
selection--multiple .select2-search--inline {
    float: right;
}
.select2-container--admin-autocomplete[dir="rtl"]select2-selection--multipleselect2-
selection__choice {
    margin-left: 5px;
    margin-right: auto;
}
.select2-container--admin-autocomplete[dir="rtl"].select2-selection--multipleselect2-
selection__choice__remove {
    margin-left: 2px;
    margin-right: auto;
}

.select2-container--admin-autocomplete.select2-container--focus .select2-selection--multiple {
    border: solid var(--body-quiet-color) 1px;
```

```
    outline: 0;
}
.select2-container--admin-autocomplete.select2-container--disabled .select2-selection--multiple {
    background-color: var(--darkened-bg);
    cursor: default;
}
.select2-container--admin-autocomplete.select2-container--disabledselect2-
selection__choice__remove {
    display: none;
}
.select2-container--admin-autocomplete.select2-container--open.select2-container--above .select2-
selection--single,select2-container--admin-autocomplete.select2-container--open.select2-
container--above .select2-selection--multiple {
    border-top-left-radius: 0;
    border-top-right-radius: 0;
}
.select2-container--admin-autocomplete.select2-container--open.select2-container--below .select2-
selection--single,.select2-container--admin-autocomplete.select2-container--open.select2-
container--below .select2-selection--multiple {
    border-bottom-left-radius: 0;
    border-bottom-right-radius: 0;
}
.select2-container--admin-autocomplete .select2-search--dropdown {
    background: var(--darkened-bg);
}

.select2-container--admin-autocomplete .select2-search--dropdown .select2-search__field {
    background: var(--body-bg);
    color: var(--body-fg);
    border: 1px solid var(--border-color);
    border-radius: 4px;
}
.select2-container--admin-autocomplete .select2-search--inline .select2-search__field {
    background: transparent;
```

```
    color: var(--body-fg);
    border: none;
    outline: 0;
    box-shadow: none;
    -webkit-appearance: textfield;
}
.select2-container--admin-autocomplete .select2-results > .select2-results__options {
    max-height: 200px;
    overflow-y: auto;
    color: var(--body-fg);
    background: var(--body-bg);
}
.select2-container--admin-autocomplete .select2-results__option[role=group] {
    padding: 0;
}
.select2-container--admin-autocomplete .select2-results__option[aria-disabled=true] {
    color: var(--body-quiet-color);
}
.select2-container--admin-autocomplete .select2-results__option[aria-selected=true] {
    background-color: var(--selected-bg);
    color: var(--body-fg);
}
.select2-container--admin-autocomplete .select2-results__option .select2-results__option {
    padding-left: 1em;
}
.select2-container--admin-autocomplete .select2-results__option .select2-results__option .select2-
results__group {
    padding-left: 0;
}
.select2-container--admin-autocomplete .select2-results__option .select2-results__option .select2-
results__option {
    margin-left: -1em;
    padding-left: 2em;
}
```

```
.select2-container--admin-autocomplete .select2-results__option .select2-results__option .select2-results__option .select2-results__option {
    margin-left: -2em;
    padding-left: 3em;
}
.select2-container--admin-autocomplete .select2-results__option .select2-results__option .select2-results__option .select2-results__option .select2-results__option {
    margin-left: -3em;
    padding-left: 4em;
}
.select2-container--admin-autocomplete .select2-results__option .select2-results__option .select2-results__option .select2-results__option .select2-results__option .select2-results__option {
    margin-left: -4em;
    padding-left: 5em;
}
.select2-container--admin-autocomplete .select2-results__option .select2-results__option .select2-results__option .select2-results__option .select2-results__option .select2-results__option .select2-results__option {
    margin-left: -5em;
    padding-left: 6em;
}
.select2-container--admin-autocomplete .select2-results__option--highlighted[aria-selected] {
    background-color: var(--primary);
    color: var(--primary-fg);
}
.select2-container--admin-autocomplete .select2-results__group {
    cursor: default;
    display: block;
    padding: 6px;
}
.errors .select2-selection {
    border: 1px solid var(--error-fg);
}
```

Java script

Autocomplete.js

```
'use strict';
{
    const $ = django.jQuery;
    $.fn.djangoAdminSelect2 = function() {
        $.each(this, function(i, element) {
            $(element).select2({
                ajax: {
                    data: (params) => {
                        return {
                            term: params.term,
                            page: params.page,
                            app_label: element.dataset.appLabel,
                            model_name: element.dataset.modelName,
                            field_name: element.dataset.fieldName
                        };
                    }
                }
            });
        });
        return this;
    };
    $(function() {
        // Initialize all autocomplete widgets except the one in the template
        // form used when a new formset is added.
        $('.admin-autocomplete').not('[name*=__prefix__]').djangoAdminSelect2();
    });
    document.addEventListener('formset:added', (event) => {
        $(event.target).find('.admin-autocomplete').djangoAdminSelect2();
    });
}
```

theme.js

```
'use strict';
{
  function setTheme(mode) {
    if (mode !== "light" && mode !== "dark" && mode !== "auto") {
      console.error(`Got invalid theme mode: ${mode}. Resetting to auto.`);
      mode = "auto";
    }
    document.documentElement.dataset.theme = mode;
    localStorage.setItem("theme", mode);
  }
  function cycleTheme() {
    const currentTheme = localStorage.getItem("theme") || "auto";
    const prefersDark = window.matchMedia("(prefers-color-scheme: dark)").matches;
    if (prefersDark) {
      // Auto (dark) -> Light -> Dark
      if (currentTheme === "auto") {
        setTheme("light");
      } else if (currentTheme === "light") {
        setTheme("dark");
      } else {
        setTheme("auto");
      }
    } else {
      // Auto (light) -> Dark -> Light
      if (currentTheme === "auto") {
        setTheme("dark");
      } else if (currentTheme === "dark") {
        setTheme("light");
      } else {
        setTheme("auto");
      }
    }
  }
```

```
    function initTheme() {
        // set theme defined in localStorage if there is one, or fallback to auto mode
        const currentTheme = localStorage.getItem("theme");
        currentTheme ? setTheme(currentTheme) : setTheme("auto");
    }
    window.addEventListener('load', function(_) {
        const buttons = document.getElementsByClassName("theme-toggle");
        Array.from(buttons).forEach((btn) => {
            btn.addEventListener("click", cycleTheme);
        });
    });
    initTheme();
}
```

BACKEND

Asgi.py

```
ASGI config for Backend project.
It exposes the ASGI callable as a module-level variable named ``application``.
For more information on this file, see
https://docs.djangoproject.com/en/5.2/howto/deployment/asgi/
"""
import os
from django.core.asgi import get_asgi_application
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Backend.settings')
application = get_asgi_application()
```

Settings.py

```
import os
from pathlib import Path
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = ["*"]
```

```
# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'Admins',
    'Users',
]
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'Backend.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
```

```
]
WSGI_APPLICATION = 'Backend.wsgi.application'
# Database
# https://docs.djangoproject.com/en/5.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
# Password validation
# https://docs.djangoproject.com/en/5.2/ref/settings/#auth-password-validators
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
# Internationalization
# https://docs.djangoproject.com/en/5.2/topics/i18n/
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True
# Static files (CSS, JavaScript, Images)
```

```
# https://docs.djangoproject.com/en/5.2/howto/static-files/
STATIC_URL = 'static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static'),]
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media'
# Default primary key field type
# https://docs.djangoproject.com/en/5.2/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

Urls.py

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from Backend.views import *
urlpatterns = [
    path('admin/', admin.site.urls),
    path('Users/', include('Users.urls')),
    path('Admins/', include('Admins.urls')),
    path('', index, name='index'),
    path('loginn/', loginn, name='loginn'),
    path('register/', register, name='register'),
    path('home_page/', index, name='home_page'),
    path('user_logout/', user_logout, name='user_logout'),
    path('user_login/', user_login, name='user_login'),
    path('user_registration/', user_registration, name='user_registration')
]
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

views.py

```
from django.shortcuts import render, redirect
from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout
from django.contrib import messages
def index(request):
    return render(request, "index.html")
def loginn(request):
    return render(request, "login.html")
def register(request):
    return render(request, "register.html")
# Define the login function
def user_login(request):
    if request.method == "POST":
        username = request.POST.get('username')
        password = request.POST.get('password')
        # Authenticate user
        user = authenticate(request, username=username, password=password)
        if user is not None:
            if not user.is_active:
                # User is inactive
                messages.error(request, "Your account is inactive. Please contact the admin.")
                return redirect('loginn')
            # Login the user
            login(request, user)
            if user.is_staff or user.is_superuser:
                # Redirect to admin home if user is staff
                return redirect('adminhome')
            else:
                # Redirect to user home if user is not staff
                return redirect('userhome')
        else:
            # Invalid username or password
            messages.error(request, "Invalid username or password.")
```

```
            return redirect('loginn')
    return render(request, 'login.html')
# Define the user registration function
def user_registration(request):
    if request.method == "POST":
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')
        confirm_password = request.POST.get('confirm_password')
        first_name = request.POST.get('first_name')
        last_name = request.POST.get('last_name')
        # Check if passwords match
        if password != confirm_password:
            messages.error(request, "Passwords do not match.")
            return redirect('register')
        # Check if username already exists
        if User.objects.filter(username=username).exists():
            messages.error(request, "Username already exists.")
            return redirect('register')
        # Check if email already exists
        if User.objects.filter(email=email).exists():
            messages.error(request, "Email already exists.")
            return redirect('register')
        # Create the user with is_active set to False
        user = User.objects.create_user(
            username=username,
            email=email,
            password=password,
            first_name=first_name,
            last_name=last_name
        )
        user.is_active = False  # Set is_active to False by default
        user.save()
        messages.success(request, "Registration successful! Please wait for admin approval.")
```

```
        return redirect('loginn')
    return render(request, 'register.html')
# Define the logout function
def user_logout(request):
    logout(request)
    messages.success(request, "You have been logged out successfully.")
    return redirect('index')
```

wsgi.py

```
import os
from django.core.wsgi import get_wsgi_application
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Backend.settings')
application = get_wsgi_application()
```

ADMINS

admin.py

```
from django.contrib import admin
```

apps.py

```
from django.apps import AppConfig
class AdminsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'Admins'
```

tests .py

```
from django.test import TestCase
```

urls.py

```
from django.urls import path
from Admins.views import *
urlpatterns = [
    path('adminhome/', adminhome, name='adminhome'),
                                path('admin_update_userstatus/<int:user_id>/',admin_update_userstatus,
name='admin_update_userstatus'),
    path('reports/', reports, name='reports'),
    path('admin_proposed/', admin_proposed, name='admin_proposed'),
]
```

Views.py

```
from django.shortcuts import render, redirect
from django.contrib.auth.models import User
from django.contrib import messages
def adminhome(request):
    users = User.objects.filter(is_staff=False, is_superuser=False)
    return render(request, "Admin/adminhome.html", {"users": users})
def admin_update_userstatus(request, user_id):
    try:
        user = User.objects.get(id=user_id)
        # Toggle the is_active status
        user.is_active = not user.is_active
        user.save()
        # Display message based on the action
        if user.is_active:
            messages.success(request, f"User {user.username} has been activated.")
        else:
            messages.success(request, f"User {user.username} has been deactivated.”)
        return redirect('adminhome')  # Redirect back to the admin home page
    except User.DoesNotExist:
        messages.error(request, "User not found.")
        return redirect('adminhome')
from Users.models import UserPrediction
def reports(request):
    # Fetch all predictions from UserPrediction table
    all_predictions = UserPrediction.objects.select_related("user").order_by("-created_at")
    return render(request, "Admin/reports.html", {"predictions": all_predictions})
def admin_proposed(request):
    return render(request, "Admin/admin_proposed.html")
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import joblib
x = df.drop('generated_power_kw', axis=1)
```

```
y = df['generated_power_kw']
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)  # Fit on all features at once
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.2, random_state=42)
# Save the scaler
joblib.dump(scaler, "scaler.pkl")
import joblib
import numpy as np
# Load the saved hybrid model
hybrid_voting = joblib.load("/content/drive/MyDrive/code/hybrid_model.pkl")
print("Hybrid Voting model loaded successfully!")
new_data =
np.array([[2.17,31,1035,0,0,0,0,0,0,0,6.37,312.71,9.36,22.62,6.62,337.62,24.48,58.753108,83.237
322,128.33543]])
scaler = joblib.load("/content/drive/MyDrive/code/scaler.pkl")
new_data_scaled = scaler.transform(new_data)
prediction = hybrid_voting.predict(new_data_scaled)
prediction = np.abs(prediction)
print("Predicted generated_power_kw:", prediction[0])
#!pip freeze > /content/drive/MyDrive/code/req.txt
```

6.2 Implementation

6.2.1 Front-End Implementation

The front-end implementation of the proposed Solar Energy Forecasting System using Machine Learning is designed to provide an intuitive, interactive, and user-friendly interface for users such as grid operators, researchers, energy analysts, and administrators. The primary objective of the front-end is to simplify user interaction with complex forecasting models and present analytical outputs in a clear and meaningful manner. It acts as the visual layer of the system, enabling users to input data, initiate forecasting processes, and interpret results without requiring technical expertise in machine learning.

The interface is developed using standard web technologies such as HTML, CSS, JavaScript, and Flask, where Flask acts as the bridge between the user interface and backend processing modules. Through this interface, users can input meteorological parameters (temperature, humidity, solar irradiance, etc.) or upload datasets for forecasting. Once the data is submitted, the system triggers the machine learning models in the backend and displays the predicted solar power output in an organized format.

The front-end also provides visualization features such as graphs, charts, and dashboards to represent forecasted solar energy trends over time. Key outputs like predicted solar power, grid stability insights, and energy dispatch recommendations are displayed using tables and visual panels, making the results easy to interpret. Additionally, the interface may support responsive design, ensuring accessibility across desktops, laptops, and mobile devices. Overall, the front-end enhances user experience by transforming complex data-driven outputs into visually understandable insights.

To enhance usability, the interface follows responsive design principles using CSS frameworks (like Bootstrap), ensuring compatibility across desktops, tablets, and mobile devices. Navigation components such as menus, forms, and buttons are designed to be simple and intuitive. Error handling and user feedback mechanisms (such as alerts and notifications) are also included to guide users during interaction.

Security and performance considerations are also incorporated in the front-end design. Input data is sanitized to prevent invalid or malicious entries, and efficient rendering techniques are used to handle large datasets. Overall, the front-end plays a crucial role in bridging the gap between users

and the forecasting system by transforming raw data and complex model outputs into meaningful visual insights.

HTML is the fundamental building block of the front-end, used to structure the content of web pages in the Solar Energy Forecasting System. It defines elements such as forms, input fields, buttons, tables, and headings that allow users to interact with the system. Through HTML, users can enter meteorological data or upload datasets, and the structure ensures that all interface components are logically organized and accessible.

CSS is used to enhance the visual appearance and layout of the web interface. It controls design aspects such as colors, fonts, spacing, alignment, and responsiveness. In this project, CSS ensures that the interface is clean, professional, and easy to navigate, while also adapting the layout for different devices like desktops, tablets, and mobile phones, improving overall user experience.

JavaScript adds interactivity and dynamic behavior to the front-end. It is used for validating user inputs, handling events (like button clicks), and updating content without reloading the page. In this system, JavaScript may also be used to send asynchronous requests to the backend (using AJAX or Fetch API) and display real-time forecasting results, making the application faster and more responsive.

Bootstrap is a popular CSS framework used to design responsive and mobile-friendly web interfaces. It provides pre-designed components such as navigation bars, forms, buttons, and grids, which help in developing a consistent and visually appealing layout quickly. In this project, Bootstrap ensures that the application works smoothly across different screen sizes and devices.

Chart.js is a JavaScript library used for data visualization in the form of charts and graphs. It helps in representing forecasted solar power data through line charts, bar graphs, and trend visualizations. This makes it easier for users to understand patterns, fluctuations, and predictions in solar energy output in a clear and interactive way.

6.2.2 Back-End Implementation

The backend forms the computational core of the Solar Energy Forecasting System, where all data processing, machine learning operations, and analytical computations are executed. It is responsible for managing the entire pipeline, from receiving user inputs to generating predictions and grid stability insights.

The backend is implemented in Python, leveraging its powerful libraries for scientific computing and machine learning. The workflow begins when input data is received through Flask APIs. This data is first passed through a data preprocessing module, which handles missing values using interpolation or imputation techniques, removes noise and outliers, and applies normalization or standardization to ensure uniform scaling.

Following preprocessing, the system performs feature engineering, where new features such as lag values (previous time-step outputs), rolling averages, and temporal features (hour of the day, day of the week, seasonal indicators) are generated. These features significantly improve the model's ability to capture temporal and nonlinear patterns in solar energy generation.

The processed data is then fed into machine learning models implemented using Scikit-learn, including Random Forest, Support Vector Regression, Decision Trees, and Gradient Boosting Regressor. Among these, Gradient Boosting is often selected due to its superior performance in handling complex nonlinear relationships and minimizing prediction errors.

The backend also includes a model evaluation module, where predictions are compared against actual values using metrics such as MAE, RMSE, and R^2 score. This ensures that only the most accurate and reliable model is used for forecasting.

A key addition to this system is the grid stability analysis module, which evaluates predicted solar output in the context of power demand. This module identifies potential fluctuations, detects risks of undergeneration or overgeneration, and provides recommendations for balancing supply and demand. It may also simulate scenarios where backup power sources or energy storage systems are required.

The backend stores all processed data, predictions, and analytical results in a SQLite3 database, enabling efficient data retrieval and historical analysis. Flask-based REST APIs are used to facilitate communication between the frontend and backend, ensuring modularity and scalability.

After preprocessing, feature engineering is performed to enhance model performance. This includes generating lag features (previous time-step values), rolling averages, and temporal features such as hour of the day, day of the week, and seasonal indicators. These features help the model capture time-dependent patterns and improve prediction accuracy for solar energy generation.

To ensure reliability, the backend includes a model evaluation module that compares predicted outputs with actual values using performance metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R^2 score. This step helps in selecting the most accurate model for deployment. An additional component of the backend is the grid stability analysis module, which evaluates predicted solar output against power demand. It identifies potential risks such as undergeneration or overgeneration and provides recommendations for balancing supply. This module may also simulate scenarios involving backup power sources or energy storage systems to maintain grid stability.

Python is the primary programming language used for backend development due to its simplicity and powerful ecosystem of libraries for data analysis and machine learning. It enables efficient implementation of data preprocessing, feature engineering, and predictive modeling. Python's flexibility allows seamless integration with web frameworks and databases, making it ideal for building scalable forecasting systems.

Scikit-learn is a widely used machine learning library in Python that provides efficient tools for building predictive models. It includes algorithms like Random Forest, Support Vector Machines, Decision Trees, and Gradient Boosting. In this project, it is used to train, test, and evaluate models for accurate solar power forecasting.

SQLite is a lightweight, serverless database used for storing input data, processed datasets, prediction results, and historical records. It enables efficient data retrieval and management without requiring a complex database setup. SQLite is particularly suitable for small to medium-scale applications due to its simplicity and reliability.

6.2.3 Deployment and Reliability

The system is designed to be flexible in deployment, supporting both local and cloud-based environments. For high-performance applications, deployment on cloud platforms such as AWS, Google Cloud Platform, or Microsoft Azure is preferred, as these platforms provide scalable

computing resources, GPU acceleration, and reliable storage solutions.In cloud deployment, the application can be hosted using services like:

- Virtual machines (EC2 instances)
- Containerized environments (Docker)
- Serverless architectures

For academic or small-scale implementations, the system can be deployed locally using Flask's development server, making it accessible through a web browser.

Reliability is ensured through multiple strategies. The system includes robust error handling mechanisms to manage issues such as invalid inputs, missing data, or model failures. Input validation ensures that only valid data is processed, while logging mechanisms track system performance and errors for debugging and monitoring.To maintain prediction accuracy over time, the system supports continuous learning, where models can be retrained periodically using updated datasets. This ensures adaptability to changing weather patterns and seasonal variations.The modular architecture of the system enhances maintainability and scalability. Individual components such as preprocessing, model training, and grid analysis can be updated independently without affecting the overall system. This also allows easy integration of advanced technologies such as deep learning models, IoT-based real-time data collection, and smart grid systems.

The implementation of the Solar Energy Forecasting System using Machine Learning provides a comprehensive solution for accurate prediction and efficient grid management. By integrating advanced machine learning techniques with real-time data processing and grid stability analysis, the system ensures reliable and sustainable energy utilization.

The combination of a user-friendly front-end, a robust backend, and scalable deployment options makes the system suitable for real-world applications in smart grids and renewable energy systems. Furthermore, its modular design allows future enhancements such as deep learning integration, IoT-based monitoring, and automated energy optimization, making it a powerful tool for next-generation energy management.

Reliability of the system is ensured through multiple mechanisms. Robust error handling techniques are implemented to manage issues such as invalid inputs, missing data, or model failures. Input validation ensures that only correct and meaningful data is processed, while logging

mechanisms continuously monitor system performance and record errors for debugging and maintenance. These features contribute to a stable and fault-tolerant system.

To maintain high prediction accuracy over time, the system supports continuous learning, where machine learning models are periodically retrained using updated datasets. This enables the system to adapt to changing weather patterns, seasonal variations, and evolving environmental conditions, thereby improving long-term performance.

The modular architecture of the system enhances both maintainability and scalability. Individual components such as data preprocessing, model training, and grid stability analysis can be updated independently without affecting the entire system. This design also allows seamless integration of advanced technologies such as deep learning models, IoT-based real-time data collection, and smart grid systems.

Overall, the implementation of the Solar Energy Forecasting System using Machine Learning provides a comprehensive and reliable solution for accurate prediction and efficient energy management. The combination of a user-friendly front-end, a robust backend, and flexible deployment options makes the system suitable for real-world applications in renewable energy and smart grid environments. Additionally, its extensible design supports future enhancements such as deep learning integration, IoT-based monitoring, and automated energy optimization, making it a powerful tool for next-generation energy systems.

7. SYSTEM TESTING

System testing is a critical phase in the Software Development Life Cycle (SDLC), as it ensures that the complete and integrated solar energy forecasting system operates correctly according to the specified requirements. This phase is performed after the successful completion of integration testing and before deployment or user acceptance testing. At this stage, the system is treated as a fully developed product, and all its components are tested together in a real or simulated environment. The primary objective of system testing is to verify whether the application performs accurately under real-world conditions and delivers reliable forecasting results along with meaningful grid stability insights.

In system testing, the application is evaluated as a whole rather than focusing on individual modules. This involves validating the interaction between various system components, such as data input, preprocessing, machine learning prediction, grid stability analysis, and output visualisation. The testing process ensures that all modules work in coordination without errors and that the system meets key quality attributes, including functionality, performance, reliability, and usability. The system is tested under different scenarios such as normal operating conditions, extreme weather inputs, and incomplete or noisy datasets to evaluate its robustness and fault tolerance. This comprehensive approach helps identify potential issues that may not be visible during unit or integration testing.

In the context of the Solar Energy Forecasting System using Machine Learning with Grid Stability, system testing plays a vital role due to the integration of multiple technologies such as data processing, predictive modelling, and energy management logic. The system must be tested to ensure that input meteorological data is correctly processed, machine learning models generate accurate solar power predictions, and grid stability analysis provides reliable recommendations for maintaining supply-demand balance. It is also essential to evaluate the system's performance under varying environmental conditions, such as changes in solar irradiance, temperature fluctuations, and seasonal variations.

Additionally, system testing verifies whether the front-end interface correctly communicates with the backend services, ensuring smooth data flow and proper visualization of results. The generated outputs, including forecasted solar energy values, graphical representations, and grid stability insights, are checked for correctness and clarity.

7.1 Types of System Testing

7.1.1 Unit Testing

Unit testing involves the design and execution of test cases to validate that individual components of the software system function correctly. It focuses on verifying the internal program logic, ensuring that inputs produce the expected outputs, and that all decision branches and execution paths are properly tested. Unit testing is performed after the completion of each individual module and before integration with other components.

In the proposed Solar Energy Forecasting System, unit testing is applied to modules such as data preprocessing, feature extraction, machine learning prediction functions, and grid stability calculations. Each module is tested independently using predefined inputs and expected outputs to ensure correctness.

Unit testing is considered a structural testing technique, as it requires knowledge of the internal code structure. It helps in identifying logical errors, incorrect computations, and coding issues at an early stage. By validating each component separately, unit testing ensures that all individual processes perform accurately according to specifications, thereby forming a strong foundation for the overall system.

7.1.2 Integration Testing

Integration testing is designed to verify that multiple software components work together correctly as a unified system. After individual modules are successfully tested through unit testing, they are combined and tested to ensure proper interaction and data flow between them.

In the Solar Energy Forecasting System, integration testing focuses on verifying the communication between modules such as data input, preprocessing, machine learning models, grid stability analysis, and output generation. Even if each module functions correctly in isolation, issues may arise when they are integrated, such as data format mismatches, incorrect parameter passing, or communication failures.

Integration testing is event-driven and emphasizes the overall behavior of the system when components interact. It ensures that the combined modules produce consistent and correct results, thereby improving system reliability and stability.

7.1.3 Functional Testing

Functional testing is performed to ensure that the system operates according to the specified functional requirements. It validates whether each function of the application behaves correctly when provided with different inputs.

This testing focuses on the following aspects:

- **Valid Input:** The system must accept all defined valid inputs and process them correctly.
- **Invalid Input:** The system must reject or handle invalid inputs appropriately without failure.
- **Functions:** All defined system functions must be executed and verified.
- **Output:** The system must generate accurate and expected outputs.
- **System/Procedures:** Interactions with other systems or processes must function correctly.

In the proposed system, functional testing includes verifying whether meteorological data is correctly processed, accurate solar forecasts are generated, and grid stability insights are produced. Testing is performed using various scenarios, including normal, boundary, and invalid conditions.Functional testing ensures that the system meets business and technical requirements and behaves correctly from the user's perspective.

7.1.4 System Testing

System testing evaluates the complete and fully integrated software system to ensure that it meets all specified requirements. It focuses on validating the overall functionality, performance, and behavior of the system in a realistic environment.

In the Solar Energy Forecasting System, system testing verifies that all modules—from data input and preprocessing to prediction, grid analysis, and visualization—work together seamlessly. It ensures that the system produces accurate forecasts, handles different input conditions, and performs efficiently.

System testing is based on real-world scenarios and process flows, emphasizing end-to-end system behavior. It ensures that the application is stable, reliable, and ready for deployment.

7.1.5 White Box Testing

White box testing is a testing technique in which the internal structure, logic, and code of the software are examined. The tester has knowledge of the internal implementation and tests the system by analysing control flow, loops, conditions, and algorithms.

In the Solar Energy Forecasting System, white box testing is applied to internal modules such as preprocessing logic, feature engineering, machine learning model implementation, and grid stability computations. This testing helps identify hidden errors, inefficient code, and logical flaws that may not be visible during external testing.

The primary objective of white box testing is to ensure that all possible execution paths are tested and that the system's internal logic is correct and efficient. It improves code quality and supports effective debugging.

7.1.6 Black Box Testing

Black box testing is a method in which the system is tested without any knowledge of its internal structure or implementation. The tester focuses only on inputs and outputs to verify whether the system behaves as expected.

In this system, black box testing involves providing different types of input data, such as meteorological parameters or datasets, and verifying whether the system generates accurate solar power predictions and grid stability insights.

The system is treated as a "black box," where the internal workings are not visible. The tester evaluates the correctness of outputs based on given inputs and requirements. This approach ensures that the system meets functional requirements and performs reliably in real-world usage scenarios.The testing process includes checking system behavior under different conditions, such as valid inputs, invalid inputs, missing data, and extreme values. Techniques such as equivalence partitioning and boundary value analysis are used to design effective test cases. Black box testing ensures that the system meets user expectations, produces correct results, and handles errors gracefully. It is particularly useful for validating system functionality from the end-user's perspective and ensuring that the application is user-friendly, reliable, and suitable for real-world deployment.

7.2 Testing Strategies

A structured testing strategy was followed throughout the development lifecycle of the proposed Solar Energy Forecasting System using Machine Learning with Grid Stability to ensure that each component of the system functioned correctly and contributed effectively to the overall performance. The testing process was carried out in a phased and systematic manner, beginning with the validation of individual modules such as data collection, preprocessing, feature engineering, machine learning model training, prediction generation, and grid stability analysis. At this stage, each module was tested independently to confirm that it produced accurate and reliable results under different input conditions. For instance, the preprocessing module was tested to ensure proper handling of missing values, normalization of data, and correct transformation of meteorological inputs, while the machine learning models were evaluated for their ability to learn patterns and generate precise solar energy forecasts. Similarly, the grid stability module was tested to verify correct computation of stability indicators based on predicted energy values. This initial phase of testing helped in identifying and resolving functional and logical errors at an early stage, thereby improving the reliability of individual system components.

Following the successful validation of individual modules, the testing strategy progressed to integration testing and full system evaluation, where all components were combined and tested as a unified pipeline. This phase focused on verifying whether the modules interacted seamlessly and whether the system could process input data efficiently from initial acquisition to final output generation. Special attention was given to ensuring the accuracy of energy predictions, consistency in data flow between modules, and correctness of grid stability analysis results. The system was further evaluated under different environmental conditions such as variations in temperature, solar irradiance, humidity, seasonal changes, and incomplete or noisy datasets to ensure robustness and adaptability. Performance aspects, including response time and processing efficiency, were also analyzed to confirm that the system could handle large datasets and real-time forecasting scenarios without degradation. By adopting this comprehensive and layered testing strategy, the system achieved improved stability, minimized the risk of unexpected failures, and ensured high confidence in its practical applicability for real-world solar energy management and grid stability optimization.

7.2.1 Test Strategy and Approach:

The testing strategy for the proposed Solar Energy Forecasting System using Machine Learning with Grid Stability was designed to ensure accurate, efficient, and reliable system performance under different conditions. Both manual and automated testing techniques were used to evaluate the performance of machine learning models, data processing, and overall system functionality. Meteorological datasets such as temperature, solar irradiance, humidity, and wind speed were used to test data preprocessing, prediction accuracy, and grid stability analysis. The strategy focused on verifying that the complete workflow—from data input to final output—operated smoothly and produced consistent and reliable results.

The primary strategic objectives included:

● Verifying correct preprocessing of meteorological data, including handling of missing values, normalization, and feature transformation before model training and prediction.

● Ensuring accurate solar energy forecasting using machine learning models by validating prediction outputs against expected or historical values.

● Validating the correctness of grid stability analysis based on predicted energy values and system parameters.

● Ensuring smooth processing of data pipelines without delays or computational inefficiencies, especially for large datasets and real-time inputs.

● Verifying proper integration between data input modules, preprocessing components, machine learning models, and output visualization modules.

● Evaluating system performance under different environmental conditions such as seasonal variations, fluctuating weather patterns, and incomplete datasets.

● Confirming that the final output, including forecast values and stability indicators, is accurate, consistent, and presented in a clear and understandable format.

7.2.2 Test Objectives:

The testing objectives were defined to ensure that all major functionalities of the proposed Solar Energy Forecasting System using Machine Learning with Grid Stability operated correctly

and met the expected performance requirements. These objectives focused on validating the accuracy of data preprocessing, prediction performance of machine learning models, correctness of grid stability analysis, and efficiency of the overall system workflow. Since the system is designed for real-world energy forecasting applications, testing also aimed to evaluate whether the model performed consistently under varying environmental conditions such as changes in temperature, solar irradiance, humidity, and seasonal variations. These objectives guided the testing process and ensured that the system delivered reliable, accurate, and practical results for energy management.
The following objectives guided the testing process:

- Ensure meteorological data is correctly processed and preprocessed.
- Verify that machine learning models accurately predict solar energy output.
- Confirm correct computation of grid stability indicators.
- Ensure the system performs effectively under different environmental conditions.
- Validate that the system processes data efficiently without significant delay.
- Ensure that prediction results and stability outputs are correctly displayed.

7.2.3 Features Tested:

testing phase covered the major functional features of the Solar Energy Forecasting System to confirm that each component was working correctly and contributing to the overall performance of the application. Features related to data preprocessing, feature engineering, machine learning prediction, grid stability analysis, and output visualisation were thoroughly examined. The purpose of this testing was to verify that the system could accurately process meteorological data and generate meaningful forecasting results under real-world conditions. By evaluating these features, the system's analytical accuracy, efficiency, and practical usability were effectively assessed.
The major system features examined during testing included:

- Correct preprocessing of meteorological data.
- Accurate solar energy prediction using machine learning models.
- Correct computation and representation of grid stability metrics.
- Reliable processing of continuous or large datasets.

7.2.4 Integration Testing Strategy:

The integration testing strategy was used to verify whether all major modules of the Solar Energy Forecasting System worked together effectively after individual testing was completed. Since the system includes multiple interconnected components such as the data input module, preprocessing pipeline, machine learning models, and grid stability analysis module, it was necessary to test their interaction and data flow. This strategy ensured that communication between modules remained accurate and consistent throughout execution. It also helped identify interface mismatches, data inconsistencies, and integration issues early, thereby improving the reliability and performance of the complete system.

- Interaction between the data input module and the preprocessing pipeline.
- Synchronisation between preprocessing outputs and machine learning model inputs.
- Compatibility between the prediction module and the grid stability analysis.
- This approach ensured smooth data flow and stable system performance.

7.2.5 Acceptance Criteria:

The acceptance criteria were defined to determine whether the proposed Solar Energy Forecasting System had successfully achieved its intended functional and performance objectives. The system was considered acceptable only when it consistently produced accurate predictions, efficient data processing, and reliable grid stability insights without major errors or inconsistencies. These criteria were based on project requirements and the system's applicability in real-world energy forecasting scenarios. Meeting these conditions ensured that the system was ready for deployment, evaluation, and further enhancement.
The system was accepted only when the following conditions were satisfied:

- Accurate prediction of solar energy based on input data.
- Stable and efficient processing of datasets without delays.
- Correct computation of grid stability indicators.
- Consistent performance under varying environmental conditions.
- Compliance with defined project requirements and functional goals

Overall Test Results

The overall testing of the Solar Energy Forecasting System using Machine Learning with Grid Stability was successfully completed, and the system demonstrated reliable and accurate performance across all testing phases. All modules, including data preprocessing, feature engineering, machine learning prediction, and grid stability analysis, were thoroughly tested both individually and in an integrated environment. The results confirmed that each component functioned correctly and contributed effectively to the overall system workflow.

During testing, the system was evaluated using various meteorological datasets under different environmental conditions to ensure robustness and adaptability. The machine learning models produced accurate solar energy forecasts, and the grid stability module generated consistent and correct stability indicators based on the predicted values. The system also handled variations in input data, including incomplete and noisy datasets, without significant degradation in performance.

No critical defects or failures were encountered during the testing process, and only minor issues, if any, were identified and resolved during earlier stages of development. Overall, the system met all functional and performance requirements defined for the project. These results indicate that the proposed system is stable, reliable, and suitable for real-world deployment in solar energy forecasting and grid stability management applications. Throughout the testing phase, no critical defects or system failures were encountered. Minor issues identified during early testing stages were resolved successfully, contributing to improved system stability. The system also demonstrated robustness by maintaining consistent performance under varying environmental conditions and input scenarios. Overall, the testing results confirm that the proposed system meets all defined functional and non-functional requirements.

In conclusion, the Solar Energy Forecasting System with Grid Stability has proven to be stable, reliable, and effective for real-world applications.

7.3 Sample Test Cases

S.No.	Test Case	Expected Result	Result	Remarks
1.	User Registration	System successfully registers a new user with valid details	Pass	Verify proper email format and password validation
2.	User Login	Registered user successfully logs into the system	Pass	Test with valid login credentials
3.	Solar Energy Prediction	System predicts solar energy output using input weather parameters	Pass	Check prediction accuracy with multiple inputs
4.	Prediction History	System stores and displays previous prediction records	Pass	Ensure data is retrieved from database correctly
5.	Admin User Management	Admin can view and manage registered users	Pass	Verify activate and deactivate user functionality
6.	Invalid username	User successfully login to the system	Fail	Test with invalid login credentials
7.	Password Mismatch Validation	User successfully registers a new user with valid details	Fail	Invalid password entered

Table no. 7.3 Test Cases

Test Case 1: User Registration

The user registration test case verifies whether a new user can successfully create an account in the solar energy forecasting system. During the registration process, the user is required to provide essential information such as first name, last name, username, email address, and password. The system must ensure that all required fields are properly filled before allowing the user to submit the registration form.

Once the user submits the form, the system performs several validation checks to confirm the correctness of the entered information. It verifies the email format, checks the strength of the password, and ensures that the username or email address has not already been registered by another user. These validation steps are important to prevent duplicate accounts and incorrect data from being stored in the system.

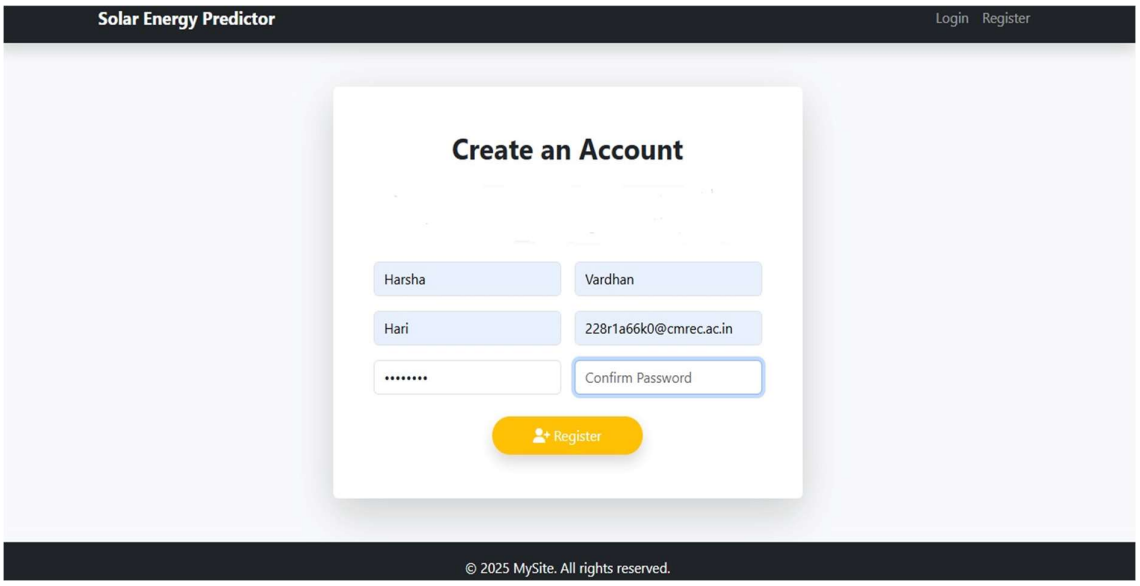

Fig 7.3.1: User Registration

Test Case 2: User Login

The user login test case verifies whether registered users can access the solar energy forecasting system using their credentials. After completing the registration process, users can log in by entering their username and password on the login page. The system must validate these credentials before granting access.

When the login request is submitted, the system compares the entered credentials with the stored data in the database. If the username and password match the stored records, the system authenticates the user and redirects them to the dashboard or main interface of the system. This allows the user to access system features and interact with the forecasting platform.

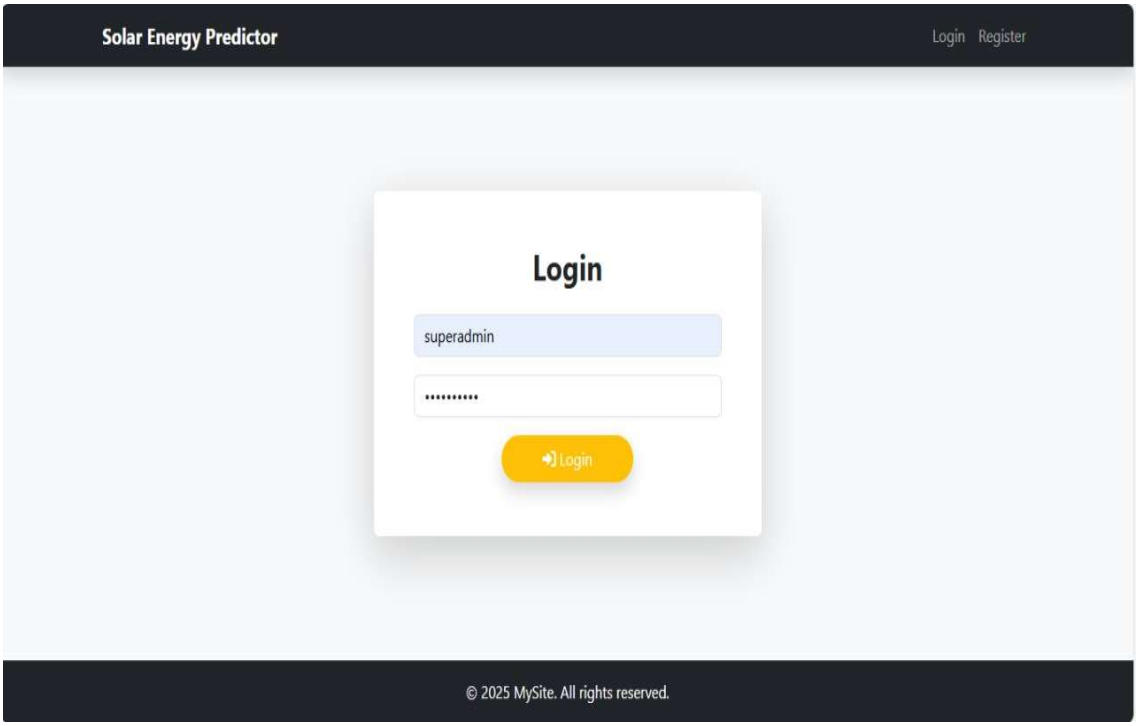

Fig 7.3.2: User Login

Test Case 3: Solar Energy Prediction

The solar energy prediction test case evaluates the core functionality of the system, which is forecasting solar energy output using machine learning techniques. The system accepts environmental input parameters such as temperature, humidity, cloud cover, solar radiation, and wind speed.

These input parameters are processed by a trained machine learning model that analyzes patterns from historical solar energy and weather data. Based on this analysis, the model predicts the expected solar power generation for a given time period.

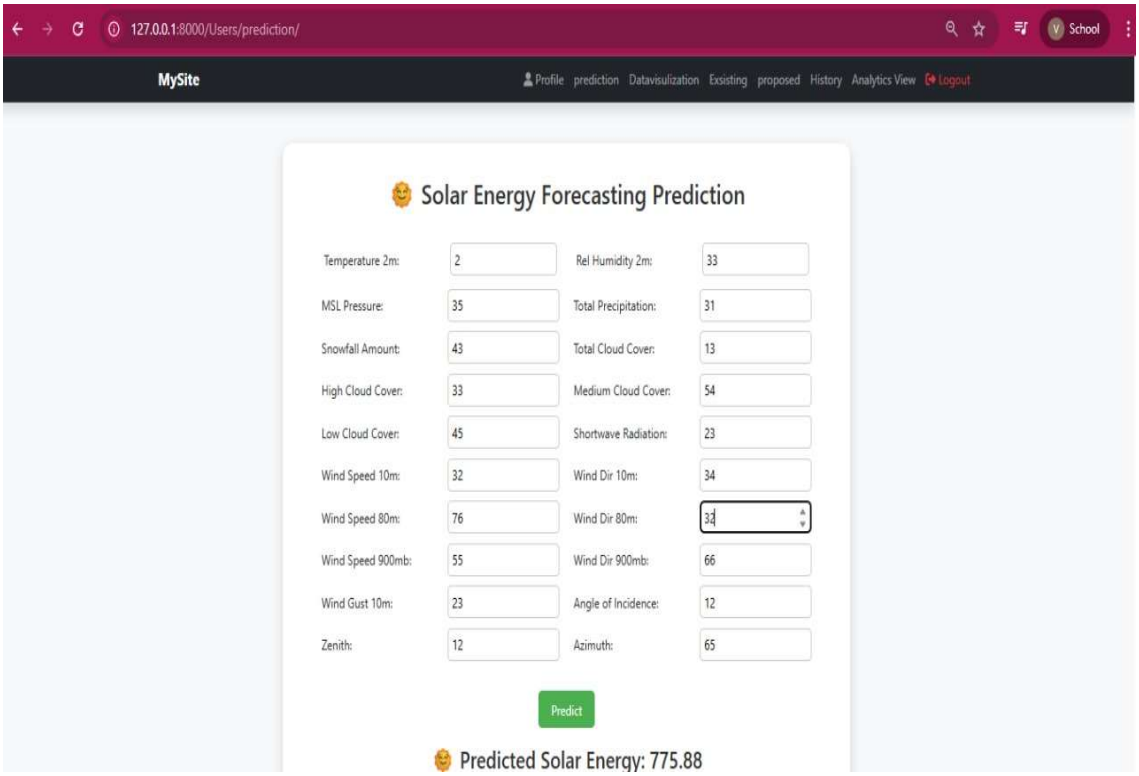

Fig 7.3.3 : Solar Energy Prediction

Test Case 4: Prediction History

The prediction history test case verifies whether the system properly stores and retrieves previous solar energy prediction results. Each time a user performs a prediction, the system records the input parameters and predicted output in the database.

Users can later access the prediction history page to view previously generated forecasts. This feature helps users analyze trends and compare past prediction results with new predictions.

Maintaining a history of predictions improves data analysis capabilities and allows users to evaluate forecasting accuracy over time. This test case ensures that prediction data is stored securely and retrieved accurately when requested.

Prediction History

Temperature 2m	Rel Humidity 2m	MSL Pressure	Total Precipitation	Snowfall Amount	Total Cloud Cover	High Cloud Cover	Medium Cloud Cover	Low Cloud Cover	Shortwave Radiation	Wind Speed 10m	Wind Dir 10m	Wind Speed 80m	Wind Dir 80m	Wind Speed 900mb	Wind Dir 900mb	Wind Gust 10m	Angle of Incidence	Zenith	Azimuth
2.17	31.0	1035.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.37	312.71	9.36	22.62	6.62	337.62	24.48	58.75	83.23	128.33
40.0	43.0	34.0	45.0	0.0	3.0	0.0	0.0	4.0	45.0	20.0	2.0	0.0	0.0	0.0	0.0	10.0	90.0	90.0	90.0
0.19	0.14	0.21	0.15	-0.02	-0.01	-0.01	-0.01	0.1	0.08	0.15	0.14	0.0	0.0	0.0	0.0	0.14	0.1	90.0	90.0
2.0	45.0	21.0	54.0	23.0	65.0	54.0	81.0	45.0	56.0	45.0	42.0	76.0	32.0	65.0	22.0	34.0	45.0	45.0	45.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
0.15	0.07	0.1	0.05	0.11	0.17	-0.03	0.04	0.24	0.05	0.44	0.03	0.14	8.0	0.12	80.0	0.15	0.5	0.17	0.1
0.02	0.03	-0.01	-0.02	0.03	0.01	0.03	0.04	-0.01	-0.03	0.03	0.01	-0.02	0.06	0.04	-0.02	0.02	0.03	-0.01	0.03

Fig 7.3.4: Prediction History

Test Case 5: Admin User Management

The Admin User Management test case verifies the functionality of the administrative module in the solar energy forecasting system. The admin panel provides administrators with the ability to monitor and control user accounts within the system. This module helps ensure that the platform operates smoothly and that all registered users are properly managed.

Through the admin interface, the administrator can view the list of registered users along with basic details such as username, email address, and account status. The administrator can also manage user accounts by activating or deactivating them when necessary. This helps maintain control over who is allowed to access the system.

This functionality is important for maintaining system security and proper administration. By managing user accounts, the administrator ensures that only authorized users can access the platform and its features. It also helps maintain the integrity and reliability of the solar energy forecasting system.

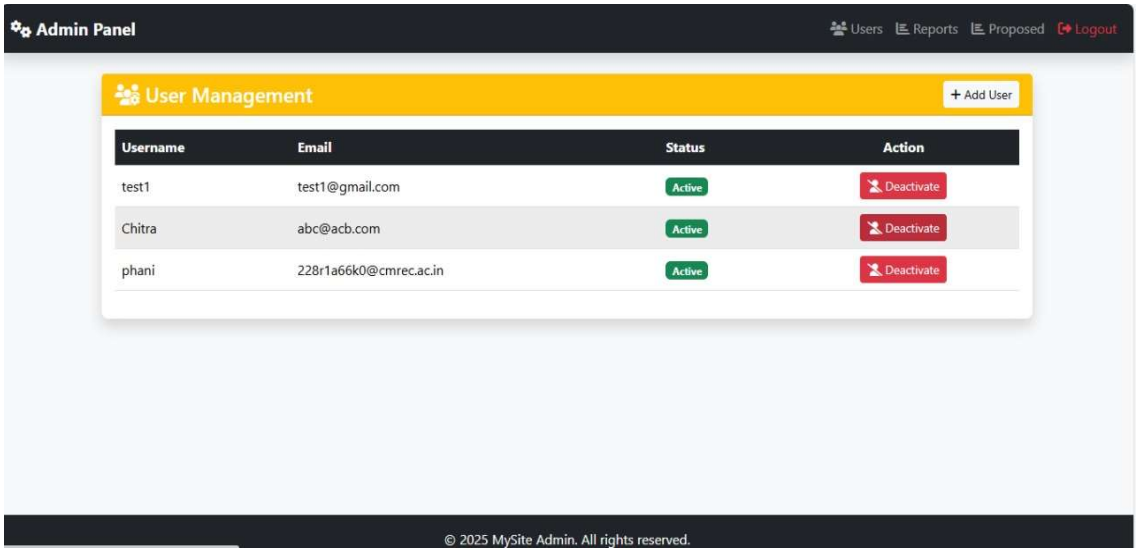

Fig 7.3.5: Admin User Management

Test Case 6: Invalid Username

The Invalid Username test case verifies the functionality of the login module in the solar energy forecasting system when incorrect credentials are entered. This validation ensures that the system properly detects and handles login attempts made with invalid usernames or incorrect passwords. It prevents unauthorized access by rejecting incorrect authentication details and displaying an appropriate error message to the user.

This functionality is essential for maintaining the security and integrity of user accounts within the system. By enforcing strict validation of login credentials, the system ensures that only authorized users can access sensitive data and forecasting features. Additionally, it enhances user experience by providing clear feedback, such as “Invalid username or password,” helping users understand the issue and retry with correct credentials. This validation also reduces potential security risks such as unauthorized access and protects the system from malicious login attempts, thereby ensuring a reliable and secure solar energy forecasting platform.

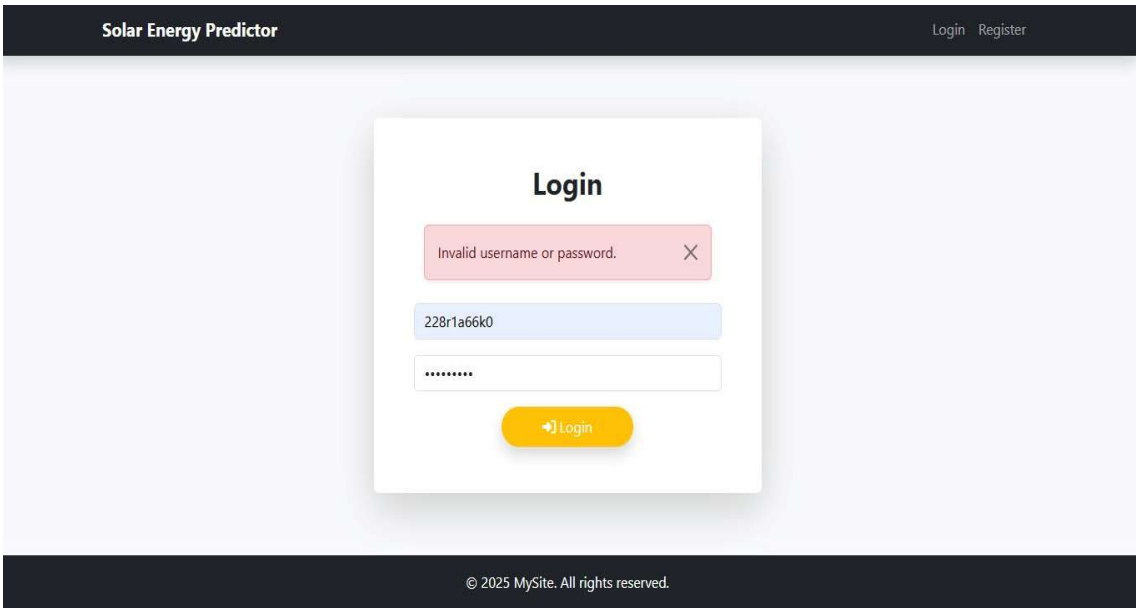

Fig 7.3.6: Invalid Username

Test Case 7: Password Mismatch Validation

The Password Mismatch Validation test case verifies the functionality of the user registration and password management modules in the solar energy forecasting system. This validation ensures that users are required to enter matching passwords during account creation or password change operations. It helps prevent accidental password errors, which could lead to user login failures or security vulnerabilities.

This functionality is critical for maintaining user account security and system integrity. By enforcing password match validation, the system ensures that users are able to access their accounts securely without accidental errors. It also reduces support issues related to forgotten or incorrectly entered passwords, thereby improving overall reliability and user experience of the solar energy forecasting platform.

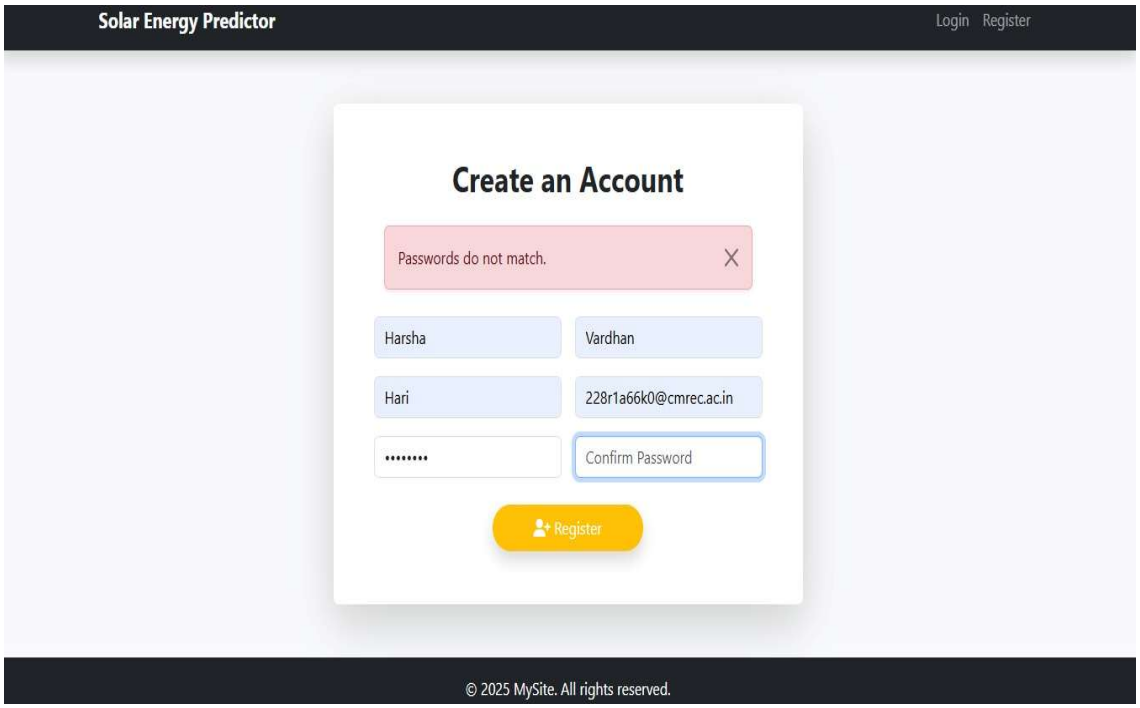

Fig 7.3.7: Password Mismatch Validation

8 . RESULTS

Fig 8.1: Home page– Solar Energy Predictor

Description: The Solar Energy Forecasting Using Machine Learning Techniques for Enhanced Grid Stability system is designed to predict solar power generation using historical and real-time weather data. The homepage of the Solar Energy Predictor provides an overview of the system, explaining how machine learning techniques analyze environmental parameters such as temperature, humidity, cloud cover, and wind speed to generate accurate solar energy predictions. The interface includes login and registration options, allowing users to securely access the platform and utilize forecasting features.

This system helps improve grid stability by enabling better energy management and efficient integration of renewable energy into the power grid. Accurate solar forecasting allows operators to optimize energy storage, reduce reliance on fossil fuels, and ensure reliable power distribution. The user-friendly design of the webpage makes it easy for users to understand the system and access its features, making it a practical solution for sustainable energy forecasting and smart grid management

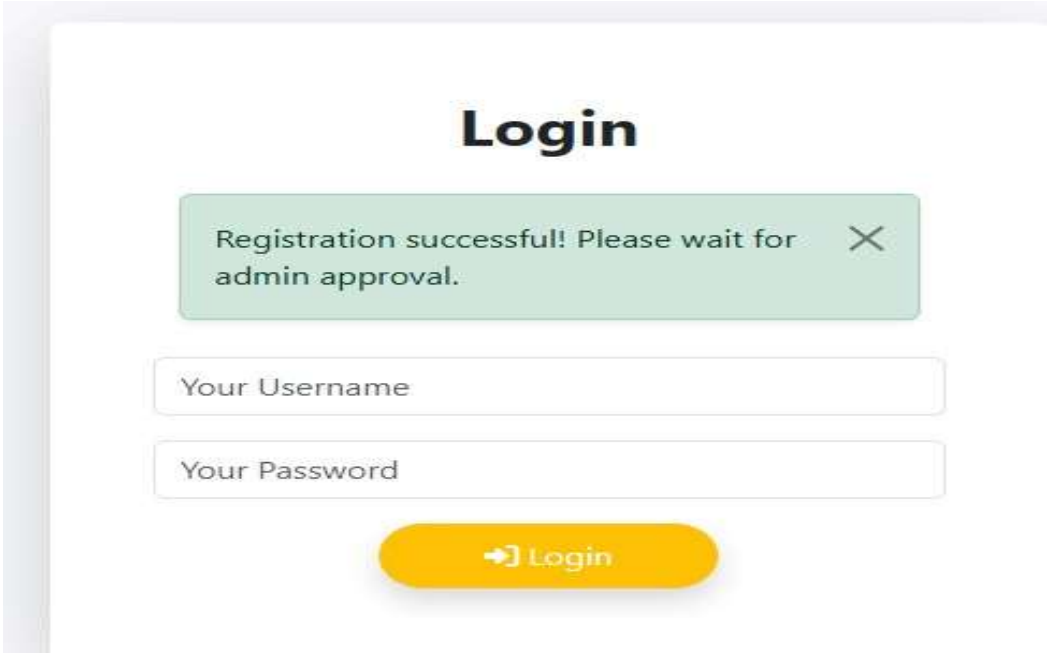

Fig 8.2 : User Login

Description: The displayed interface represents the login page of the Solar Energy Forecasting Using Machine Learning Techniques for Enhanced Grid Stability system. This page allows users to securely access the platform by entering their login credentials. The system includes navigation options such as Login and Register at the top, enabling new users to create an account and existing users to access the solar energy prediction system. The clean and structured design ensures easy interaction, while the background image related to solar energy enhances the visual appeal and clearly represents the renewable energy theme of the project.

This login interface plays an important role in maintaining system security and user authentication by allowing only authorized users to access forecasting features and prediction modules. After successful login, users can interact with the system to view solar energy forecasts, analyze weather-based predictions, and manage grid stability information. The simple and user-friendly design improves accessibility and usability, making the system efficient for energy analysts, researchers, and operators who need accurate solar energy forecasting and reliable renewable energy management.

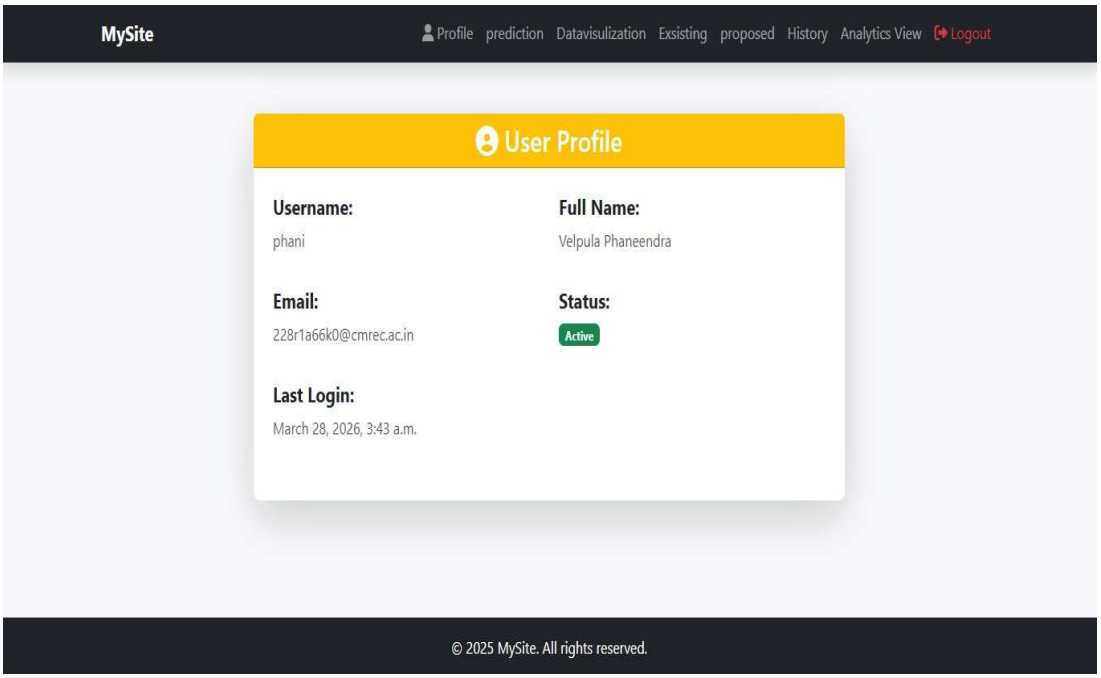

Fig 8.3 : Profile View

Description: The User Profile page of the Solar Energy Forecasting Using Machine Learning Techniques for Enhanced Grid Stability system serves as a centralized interface for managing user account information. It displays key details such as username, full name, email ID, account status, role, and last login time. This allows users to verify their identity, track account activity, and ensure that their information is accurate and up to date.

In addition to displaying user details, the module provides essential functionalities such as updating profile information, changing passwords, and managing account settings. Secure authentication mechanisms are implemented to protect user data, including password validation and controlled access. Role-based access control can also be incorporated to ensure that only authorized users can access specific features such as forecasting results and system administration tools.

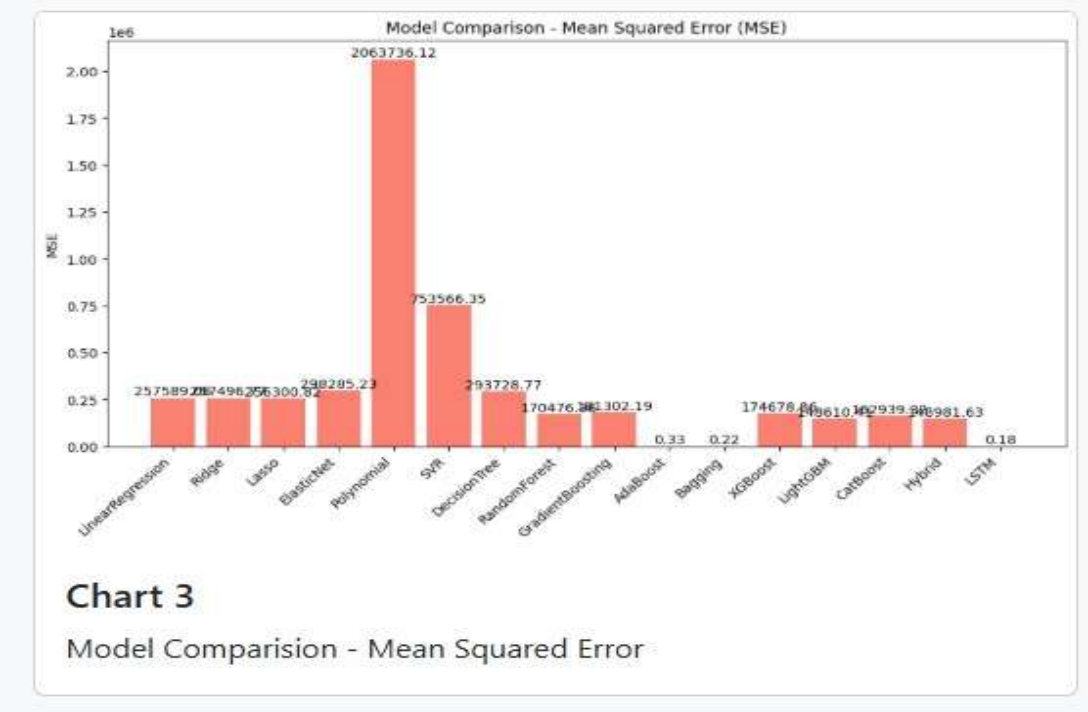

Fig 8.4 : Data Visualization

Description: The displayed interface represents the Model ComparisonMean Squared Error (MSE) of the Solar Energy Forecasting Using Machine Learning Techniques for Enhanced Grid Stability system. This chart visually compares the performance of multiple machine learning models based on their MSE values, which measure the difference between predicted and actual solar energy outputs. The graph includes models such as Linear Regression, Ridge, Lasso, ElasticNet, Polynomial Regression, SVR, Decision Tree, Random Forest, Gradient Boosting, AdaBoost, Bagging, XGBoost, LightGBM, CatBoost, Hybrid, and LSTM. The bar chart format provides a clear and structured visualization, allowing users to easily identify which models perform better, while the labeled values enhance readability and interpretation.

This comparison chart plays a crucial role in evaluating model efficiency and selecting the most accurate algorithm for solar energy prediction. By analyzing this chart, users can understand the strengths and limitations of each model, enabling better decision-making for improving forecasting accuracy and ensuring enhanced grid stability.

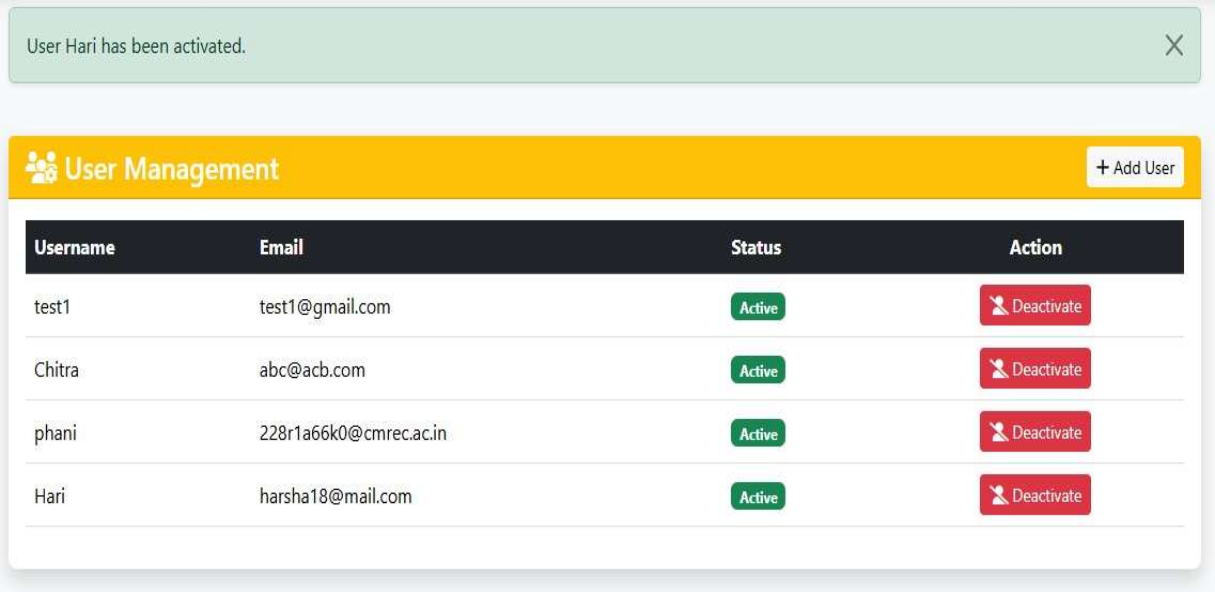

Fig 8.5: Admin User Management

Description: The displayed interface represents the User Management page of the Solar Energy Forecasting Using Machine Learning Techniques for Enhanced Grid Stability system. This page provides an organized view of all registered users, including details such as username, email ID, account status, and available actions. At the top, a notification message confirms successful user activation, enhancing system feedback and user awareness. The interface also includes an "Add User" option, allowing administrators to easily register new users. The tabular format ensures clarity and structured presentation, making it simple to monitor and manage user information efficiently.

This user management module plays a vital role in maintaining system security and access control by allowing administrators to activate or deactivate user accounts as needed. Each user entry includes a status indicator (Active) and an action button (Deactivate), enabling quick modifications to user permissions. This ensures that only authorized individuals can access the solar energy forecasting system and its analytical features. The clean and user-friendly design improves usability, making it easier for administrators to manage users, monitor system access, and ensure smooth operation of the platform for energy analysts, researchers, and system operators.

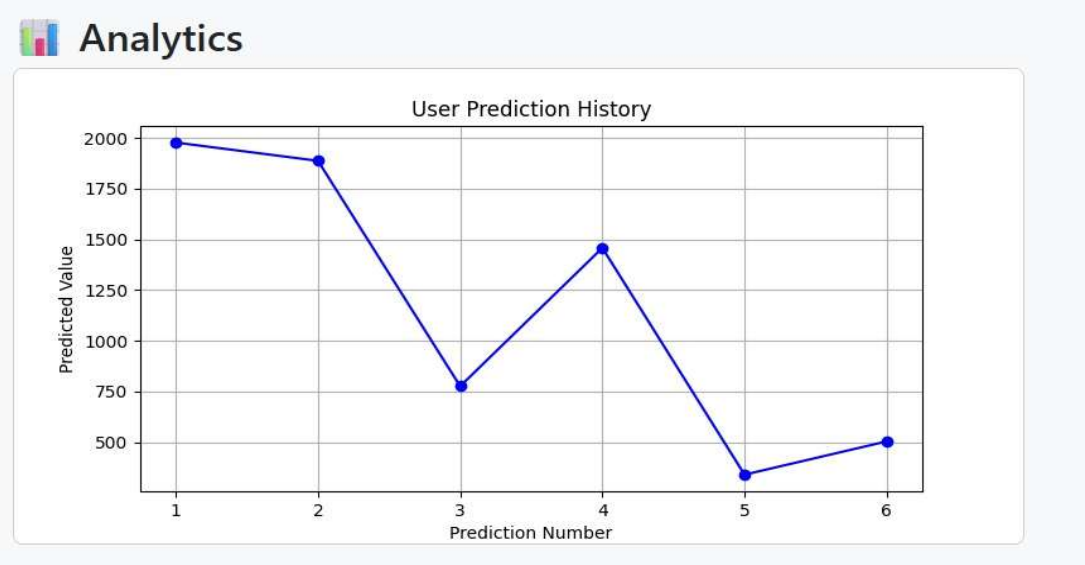

Fig 8.6: Analytics View

Description: The displayed interface represents a prediction result visualization graph in the Solar Energy Forecasting Using Machine Learning Techniques for Enhanced Grid Stability system. This graph illustrates the predicted solar energy values across different prediction instances, labeled as "Prediction Number" on the x-axis and corresponding output values on the y-axis. The plotted line with markers clearly shows the variation in predicted values, helping users understand trends and fluctuations in solar energy generation over time.

This visualization plays an important role in analyzing forecasting performance by providing a clear and interactive representation of prediction results. Users can easily observe increases or decreases in energy output, identify patterns, and evaluate the consistency of the model's predictions. Such graphical representation enhances decision-making by enabling energy analysts and system operators to interpret forecasting data quickly and accurately, ultimately supporting better grid stability and efficient renewable energy management.

9. CONCLUSION

The rapid growth of renewable energy sources has significantly increased the demand for accurate, intelligent, and real-time forecasting systems to maintain grid stability, efficiency, and reliability [1], [2]. Among various renewable sources, solar energy has emerged as one of the most promising due to its sustainability, abundance, and environmentally friendly nature. However, solar power generation is inherently variable and highly dependent on dynamic environmental conditions such as solar irradiance, temperature, humidity, wind speed, and cloud cover. These fluctuations introduce uncertainty into power generation, making accurate prediction a complex and challenging task. Hence, the development of efficient and reliable forecasting systems is crucial for optimal energy utilization, grid balancing, and minimizing energy losses [3].

In this project, a Solar Energy Forecasting System based on Machine Learning techniques was developed to address these challenges effectively. The system utilizes historical meteorological and solar generation data to learn underlying patterns and correlations between weather parameters and energy output. Various machine learning models were implemented and evaluated to identify the most suitable algorithm for accurate prediction. The workflow includes key stages such as data collection, preprocessing (handling missing values, normalization), feature selection, model training, testing, and performance evaluation using metrics like Mean Squared Error (MSE). Additionally, visualization modules were incorporated to present prediction results in a clear and interpretable manner [4]. The system demonstrated strong predictive performance, consistency across datasets, and the ability to generalize well under varying environmental conditions.

Furthermore, compared to traditional statistical and time-series forecasting methods, the proposed machine learning-based approach offers improved accuracy, adaptability, and efficiency. Advanced models such as ensemble learning techniques and deep learning models (e.g., LSTM) showed superior performance in capturing complex nonlinear relationships in the data. This enhances the reliability of predictions and supports critical applications such as load forecasting, energy scheduling, and smart grid management [1], [5]. The integration of such intelligent systems helps reduce operational costs, optimize resource allocation, and improve overall energy efficiency. Overall, the developed system clearly demonstrates that machine learning techniques can significantly enhance solar energy forecasting accuracy and contribute to better planning and

operation of modern power systems. It supports informed decision-making, reduces uncertainty, and plays a vital role in achieving sustainable energy goals. The system is especially beneficial for smart grid environments, where real-time analysis and adaptive control are essential for maintaining balance between energy supply and demand [4], [6].

In conclusion, this work highlights the effectiveness of integrating machine learning with renewable energy systems to address real-world challenges in solar power forecasting. Future enhancements can include the incorporation of real-time data streams, deployment of advanced deep learning architectures such as LSTM and GRU, hybrid model optimization, and the use of large-scale datasets for improved scalability and robustness. Additionally, integrating IoT devices and cloud-based platforms can further enhance system performance, enabling real-time monitoring and intelligent energy management in next-generation smart grids [2], [6].

10. FUTURE ENHANCEMENTS

Although the proposed Solar Energy Forecasting System provides reliable predictions using machine learning techniques, there is scope for further improvement to enhance accuracy, scalability, and real-world applicability. Future developments can focus on integrating advanced technologies to make the system more adaptive and efficient for smart grid environments.

One key enhancement is the integration of real-time weather data from meteorological services or APIs. Currently, the system relies mainly on historical data, which may limit its ability to respond to changing conditions. By incorporating real-time parameters such as solar irradiance, temperature, humidity, wind speed, and cloud cover, the system can generate more accurate and timely forecasts, improving energy management decisions.Another improvement is the use of advanced deep learning models such as Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNN). These models are well-suited for time-series data and can capture complex patterns in solar energy generation. Their implementation can enhance prediction accuracy and better handle seasonal and weather variations.

The system can also be extended using Internet of Things (IoT) technology for automated data collection. Sensors installed in solar systems can continuously provide real-time environmental data, reducing manual effort and improving system efficiency.Additionally, developing a user-friendly web or mobile interface can improve accessibility. Such platforms can help users visualize predictions, analyze trends, and monitor performance effectively.

Integration with smart grid technologies is another promising direction. This allows dynamic adjustment of power distribution based on predicted solar output, improving load balancing and reducing energy wastage.Furthermore, using larger and more diverse datasets can improve the model's performance and adaptability. Cloud computing can also be utilized to support large-scale data processing and real-time analysis.

In conclusion, by incorporating real-time data, advanced models, IoT, and smart grid integration, the system can evolve into a more accurate, scalable, and intelligent solution for renewable energy management.

REFERENCES

1. M. Abubakar, Y. Che, M. Faheem, M. S. Bhutta, and A. Q. Mudasar, "Intelligent modeling and optimization of solar plant production integration in the smart grid using machine learning models," *Advanced Energy and Sustainability Research*, vol. 5, no. 4, Apr. 2026, Art. no. 2300160.
2. M. A. Nasab, M. A. Dashtaki, B. Ehsanmaleki, M. Zand, and P. Sanjeevikumar, "LFC of smart, interconnected power system in the presence of renewable energy sources using coordinated control design of hybrid electric vehicles," *Renewable Energy Focus*, vol. 50, Sep. 2025, Art. no. 100609.
3. E. Swarnalatha, J. Suresh, M. Shirisha, and N. Madhavi, "AI Enabled Microgrid Energy Management Systems Using Machine Learning Algorithms: A Survey," *International Journal of Engineering Research and Applications*, 2024.
4. R. Tiwari, H. Nayak, and R. Reddy, "Enhanced Power Quality and Forecasting for PV-Wind Microgrid Using Neural Network-Based Time Series Forecasting," *Journal of Data Acquisition and Processing*, 2024.
5. S. R. K. Reddy, J. B. V. Subrahmanyam, and A. Srinivasula Reddy, "Voltage Control of Self Excited Induction Generator Using Artificial Neural Network," *Journal of Engineering and Technology (JETIR)*, 2023.
6. J. Gaboitaolelwe *et al.*, "Machine learning based solar photovoltaic power forecasting: A review and comparison," *IEEE Access*, vol. 11, pp. 40820–40845, 2023.
7. F. Demir and H. Citakoglu, "Forecasting of solar radiation using different machine learning approaches," *Neural Computing and Applications*, vol. 35, no. 1, pp. 887–906, Jan. 2023.
8. N. E. Benti *et al.*, "Forecasting renewable energy generation with machine learning and deep learning: Current advances and future prospects," *Sustainability*, vol. 15, Apr. 2023, Art. no. 7087.
9. A. Das *et al.*, "Hybrid deep learning architecture for short-term solar irradiance forecasting," *Energy AI*, vol. 12, 2023, Art. no. 100244.
10. M. A. Khan *et al.*, "Solar power forecasting using CNN–LSTM hybrid deep learning model," *Renewable Energy*, vol. 200, pp. 1201–1215, Feb. 2023.
11. M. Mohamed *et al.*, "Dynamic forecasting of solar energy microgrid systems using feature engineering," *IEEE Transactions on Industry Applications*, vol. 58, no. 6, pp. 7857–7869, Nov. 2022.

12. A. Bouraiou *et al.*, “Field investigation of PV pumping system ageing failures under Saharan environment,” *Solar Energy*, vol. 243, pp. 142–152, Sep. 2022.
13. P. Singh *et al.*, “Solar photovoltaic energy forecasting using machine learning and deep learning technique,” in *Proc. IEEE 9th International Conference on Power, Control and Computing Technologies (UPCON)*, Dec. 2022, pp. 1–6.
14. S. Aslam *et al.*, “A survey on deep learning methods for power load and renewable energy forecasting in smart microgrids,” *Renewable and Sustainable Energy Reviews*, vol. 144, Jul. 2021, Art. no. 110992.
15. K. K. Hiran *et al.*, *Machine Learning for Sustainable Development*, vol. 9. Berlin, Germany: Walter de Gruyter GmbH & Co. KG, 2021.
16. A. M. Alam *et al.*, “Solar PV power forecasting using traditional and machine learning techniques,” in *Proc. IEEE Kansas Power and Energy Conference*, Apr. 2021, pp. 1–5.
17. U. Munawar and Z. Wang, “A framework of using machine learning approaches for short-term solar power forecasting,” *Journal of Electrical Engineering & Technology*, vol. 15, no. 2, pp. 561–569, Mar. 2020.
18. N. Hari *et al.*, “Gallium nitride power electronic devices modeling using machine learning,” *IEEE Access*, vol. 8, pp. 119654–119667, 2020.
19. Jebli *et al.*, “The forecasting of solar energy based on machine learning,” in *Proc. International Conference on Engineering & Information Technology (ICEIT)*, Mar. 2020, pp. 1–8.
20. E. D. Obando *et al.*, “Solar radiation prediction using machine learning techniques: A review,” *IEEE Latin America Transactions*, vol. 17, no. 4, pp. 684–697, Apr. 2019.