AN INTELLIGENT ELECTRICITY MANAGEMENT UNIT: AI-DRIVEN POWER FORECASTING AND PERSONALIZED CONSUMPTION INSIGHTS WITH APPLICATION INTEGRATION

Balasuriya B.L.I.S.
IT21803666

B.Sc. (Hons) Degree in Information Technology Specializing in Information Technology

Department of Information Technology
Sri Lanka Institute of Information Technology
Sri Lanka

August 2025

FUTUREWATT USAGE PREDICTOR – A PREDICTIVE ANALYTICS APPROACH FOR HOUSEHOLD ENERGY CONSUMPTION

Balasuriya B.L.I.S.

IT21803666

The dissertation was submitted in partial fulfilment of the requirements for the Bachelor of Science (Hons) in Information Technology specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

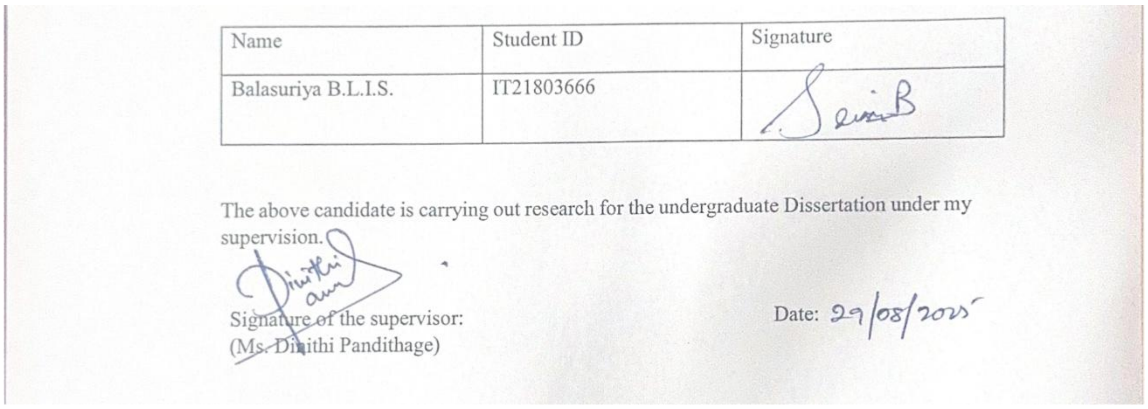
Sri Lanka

August 2025

DECLARE

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

Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).”



Name	Student ID	Signature
Balasuriya B.L.I.S.	IT21803666	[illegible]

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

[illegible]

Signature of the supervisor:
(Ms. Dinithi Pandithage)

Date: 29/08/2025

ABSTRACT

Although smart meters offer a high-level view of household energy consumption, they fail to provide the granular insights needed to understand and predict the usage of individual appliances. The goal of this project is to create an intelligent system that tracks the energy consumption of specific appliances in real time, prioritizes forecast accuracy, and makes use of machine learning to enhance energy management. The system's features include the creation of dedicated forecasting models for each appliance using high-frequency IoT data, and the ability to provide users with personalized, forward-looking consumption reports. It will learn the unique "energy fingerprint" of each device, enabling homeowners to receive accurate forecasts, identify high-consumption appliances, and, in the end, giving them the tools to better manage their electricity costs.

This project is unique in that it combines high-frequency IoT sensor data with a "specialist model" machine learning approach to create personalized and highly accurate forecasts for individual devices, rather than a single, generalized household prediction. The ultimate result will be a fully functioning predictive system, accessible via an API, that aims to enhance residential energy conservation by raising homeowners' awareness, satisfaction, and control over their electricity usage.

Keywords: IoT, Energy Forecasting, Time-Series Analysis, Machine Learning, XGBoost, Smart Home, Appliance-Level Monitoring.

.

ACKNOWLEDGEMENT

I express sincere gratitude to my dissertation supervisors, Ms. Dinithi Pandithage and Mr. Ashvinda Iddamalgoda, for their unwavering support. Thanks to the Faculty of Computing and Department staff and fellow students for their contributions, provided valuable support. I also thank my family and friends for their unwavering encouragement during this research journey.

Table of Contents

Declaration ... 3

Abstract ... 4

Acknowledgement ... 5

List of Abbreviations ... 8

1.0 INTRODUCTION ... 10

- **1.1 Background and Literature Survey** ... 10
  - **1.1.1 Background** ... 10
  - **1.1.2 Literature Review** ... 11
- **1.2 Research gap** ... 13
- **1.3 Research Problem** ... 16
- **1.4 Research Objectives** ... 18
  - **1.4.1 General Objective** ... 18
  - **1.4.2 Specific Objectives** ... 18

2.0 METHODOLOGY ... 20

- **2.1 Research Methodology Framework** ... 20
- **2.2 System Architecture Design** ... 22
- **2.3 Data Collection and Preprocessing** ... 24
  - **2.3.1 Data Collection Process** ... 24
  - **2.3.2 Temporal Indexing and Cleaning** ... 24
  - **2.3.3 Dataset splitting strategy** ... 25
- **2.4 Feature Engineering** ... 27
  - **2.4.1 Time-Based Features** ... 27
  - **2.4.2 Lag Features** ... 27
  - **2.4.3 Rolling Window Features** ... 27
- **2.5 The Specialist Forecasting Model (XGBoost)** ... 29
- **2.6 Recursive Multi-Step Forecasting Approach** ... 31
- **2.7 Monthly Energy Consumption Forecasting using SARIMA** ... 32
- **2.8 Integrated Chatbot Feature (EcoBot)** ... 34
- **2.9 Model Performance Evaluation Metrics** ... 36
  - **2.9.1 Mean Absolute Error (MAE)** ... 36

2.9.2 Mean Absolute Percentage Error (MAPE) 37
2.9.3 Visualization of Model Accuracy....37
2.10 Testing and Implementation....39
2.10.1 Testing the Forecasting Models....39
2.10.2 Usability Test and Feedback....40
2.10.3 Final Deployment....43
2.11 Commercialization Aspects of the product....44
2.11.1 Market Potential and Target Users....44
2.11.2 Product Positioning and Value Proposition....44
2.11.3 Monetization and Revenue Models 45
2.11.4 Scalability and Future Expansion 46
2.11.5 Technical Feasibility for Market Use 46
3.0 RESULTS & DISCUSSION 47
3.1 Results....47
3.2 Research findings....48
3.3 Discussion 49
4.0 CONCLUSION 50
5.0 REFERENCES....51
6.0 APPENDICES....53
Appendix A: Sample Dataset collected from the IoT Device 53
Appendix B: Model Training Code Snippets 53
B.1: Model – AC (Train/Test & Model Training)....53
B.2: Model – Laptop (Train/Test & Model Training) 55
B.3: Model – Refrigerator (Train/Test & Model Training) 56
B.4: Model – Monthly usage predictor (Train/Test & Model Training) 57
Appendix C: Flask API Endpoints....60
C.1: POST - /predict 60
C.1: GET - /Feature-importance 63
C.1: POST - /chat....63

LIST OF FIGURES

Figure 1 – System Architecture Design 22
Figure 2 - Train/Test executed for model AC 25
Figure 3 - Sample pre-processed dataset from the AC model 25
Figure 4 - Sample dataset with features created 28
Figure 5- Feature Importance graph for AC Model 28
Figure 6 - Key Hyperparameters used for AC Model 30
Figure 7 - Recursive forecasting used to predict the next 4 hours of AC consumption 31
Figure 8 - Sample of Forecasted daily kWh 33
Figure 9 - Predicted total energy consumption for next month 33
Figure 10 - UI of EcoBot 35
Figure 11-MAE Formula 36
Figure 12 - MAE and MAPE value for AC Model 36
Figure 13-MAPE Formula 37
Figure 14 - Actual vs prediction graph for AC model 38
Figure 15 - Actual vs Prediction graph for Fridge model 38
Figure 16 - UI for prediction feature 40
Figure 17- UI of the prediction output 41
Figure 18 - Key Features Section of the model 41
Figure 19 - Generated future forecast graph 42
Figure 20 - Model Validation Graph compared to last 3 hours 42
Figure 21 - Sample Dataset Snapshot 53

LIST OF TABLES

Table 1 - Summary of research gaps in existing energy prediction components 14
Table 2 - Comparison of existing research and identified gaps 15
Table 3 - Summary of dataset fields with description 26
Table 4 - Key Hyperparameters used when modelling 30
Table 5 - Summary of MAE values of the models 47

LIST OF ABBREVIATIONS

Abbreviation	Description
AI	Artificial Intelligence
API	Application Programming Interface
HEMS	Home Energy Management System
IoT	Internet of Things
kWh	Kilowatt Hour
ML	Machine Learning
SARIMA	Seasonal AutoRegressive Integrated Moving Average
S3	Amazon Simple Storage Service

1.0 INTRODUCTION

1.1 Background and Literature Survey

1.1.1 Background

Energy management is a critical intersection of technology, economics, and environmental sustainability, offering individuals an experiential way of understanding and controlling their resource consumption. Among the most impactful areas for this is residential energy monitoring, which gives homeowners insight into their daily electricity usage. Smart metering in countries across the globe has been a significant first step, generating valuable data for utility companies and providing consumers with high-level summaries of their power draw. This not only empowers consumers to be more mindful of their usage but also contributes to greater grid stability and promotes a broader commitment to energy conservation.

But with the widespread adoption of smart home technology comes a complex set of challenges. Most pressing is the lack of granularity in the data provided. Standard smart meters report on the total household consumption, leaving consumers unable to identify the specific appliances responsible for spikes in their electricity bill. A clear majority of homeowners lack the tools or expertise to effectively differentiate the consumption patterns of their devices. This ignorance can lead to lost opportunities for cost savings and inefficient energy use. For instance, misinterpreting a high bill because of lighting usage, while overlooking the continuous, high power draw of an aging refrigerator, can lead to ineffective conservation efforts. Such scenarios are common in households where multiple devices contribute to a cumulative, yet opaque, energy footprint.

Second, traditional energy monitoring is highly dependent on historical data, offering retrospective reports rather than predictive insights. Consumers can see how much energy they used *last month*, but they lack the tools to forecast how their behaviour today will impact their bill *next month*. While most utility portals offer basic summaries, this human-based analysis model lacks personalization and predictive power. Consumers who want to make informed decisions—such as when to run a high-power appliance like a washing machine—do not have the necessary forward-looking guidance for an optimized and cost-effective experience. The lack of standardized digital tools to provide real-time, appliance-specific forecasts only exacerbates this.

Such gaps can be filled by modern technological interventions, redefining how consumers interact with their energy usage. The Internet of Things (IoT) has provided an unprecedented opportunity to collect high-frequency, device-specific data from any outlet in a home. Machine Learning (ML), and particularly regression algorithms like XGBoost, has proven very promising in learning complex, non-linear patterns from time-series data, making it possible to accurately forecast future consumption. At the same time, cloud infrastructure offers a scalable solution for storing vast amounts of sensor data and deploying these trained models.

This project proposes the development of an Intelligent System for Appliance-Specific Energy Consumption Forecasting. The proposed system is intended to be useful to homeowners by providing the following essential features:

- Real-time, high-frequency data collection from a dedicated IoT device for any given appliance.
- The training of a unique "specialist model" for each appliance to learn its specific energy fingerprint.
- Generation of a high-confidence, multi-hour future forecast based on historical data and learned patterns.
- A scalable web API to serve these personalized predictions to a user-facing application.

By incorporating all these attributes, the system addresses the most critical loopholes in present energy monitoring solutions. As opposed to most smart meters that report on aggregated household data, this system is specifically tuned to provide granular, device-level predictions. While it allows for intelligent monitoring, it also functions as a predictive guide, allowing the user to make quick, fact-based decisions about their future energy usage.

By doing so, this system is set to transform energy management from passive observation to active, predictive control, empowering individuals with the information and foresight to engage responsibly and cost-effectively with their energy consumption. This innovation is part of broader trends in smart grid technology and personalized smart home systems, setting a precedent for future development in consumer-centric energy solutions.

1.1.2 Literature Review

The application of IoT for home energy management has been widely explored in recent years. Prathyusha and Bhowmik [1] presented an IoT-based smart home application focusing on energy management, describing device-level sensing and cloud integration for alerts and basic analytics. Their work emphasized practical device connectivity and a user-facing application, but it remained primarily at the systems-integration level rather than deep, per-appliance forecasting.

Bharadwaj et al. [2] reviewed advances in smart home energy monitoring, highlighting increased sensor adoption and the importance of real-time monitoring. They underscored how modern IoT setups can provide continuous visibility into household consumption but noted a gap in predictive functionality — many systems still focus on historical reporting rather than forward-looking forecasts.

Kabalci and Kabalci [9] surveyed smart monitoring infrastructures for energy-efficient IoT and discussed best practices for scalable sensor networks and data pipelines. Their analysis laid a foundation for reliable data collection and edge-to-cloud architectures, which are prerequisites for any accurate forecasting system.

At the data analytics and algorithmic layer, Deb, Zhang, and Lee [4] compared machine learning techniques for forecasting building energy consumption and found that modern ML methods can outperform traditional statistical models in capturing complex consumption patterns. Their results encourage the use of feature-rich models in energy forecasting tasks.

Machorro-Cano et al. [5] proposed HEMS-IoT, a big-data and ML-based smart home system that integrates large-scale sensor data with machine learning for energy saving. Their work demonstrated how centralized ML on aggregated data can drive savings but focused on household-level strategies rather than specialized per-device models.

Bai and Wang [6] discussed AI-powered personalized energy management systems, emphasizing the value of tailoring recommendations and predictions to individual households. They showed that personalization improves user engagement and the efficacy of energy-saving suggestions but did not deeply explore device-specific model architectures or deployment strategies for high-frequency data.

Condon et al. [7] described the design and implementation of a cloud-IoT-based home energy management system, including architectural patterns for data ingestion and storage. Their implementation-level insights are highly relevant to deploying forecasting services in production environments.

Sardianos et al. [8] investigated real-time personalized energy-saving recommendations, illustrating the role of interpretable analytics and user-facing recommendations that act on forecasted or observed behavior. Their user-centric focus highlights the need for explainability and accessible outputs — points that are central to practical adoption.

Finally, Deepa and Kodabagi [3] articulated data analytics challenges in smart grids and smart energy management, including data volume, quality, and heterogeneity. They stressed the need for robust preprocessing, scalable storage, and feature engineering to support reliable ML models in energy domains.

1.2 Research gap

Despite a healthy corpus of work on IoT-based monitoring, ML forecasting, and HEMS deployments, the reviewed literature reveals several gaps that this thesis addresses:

1. Granularity gap:

   Many existing studies and systems focus on household-level or aggregated forecasts ([5], [7]). Few solutions produce high-frequency, per-appliance forecasts that can capture device-specific behaviours.

2. Specialist-model gap:

   While ML methods are applied to energy forecasting ([4], [6]), there is limited work on the specialist model paradigm, training dedicated models per device to learn unique "energy fingerprints" and improve short-term accuracy.

3. Operationalization gap:

   Several papers describe cloud-IoT architectures ([1], [9], [7]) or large-scale analytics, but fewer demonstrate real-time model serving with lightweight APIs and recursive high-frequency forecasting in a production-like pipeline.

4. Explain ability / user interaction gap:

   User-facing interpretability and interactive explain ability (recommendation and conversational interfaces) are seldom integrated with forecasting engines. Sardianos et al. [8] advocate for personalized recommendations, but integrated conversational explain ability (a chatbot that reasons about forecasts) is still underexplored.

5. Evaluation gap:

   Comparisons between traditional statistical models and feature-rich tree-based models at high temporal resolution are limited. Many studies test on aggregated daily/hourly data, leaving high-frequency (seconds-level) evaluation underreported.

Area	**Findings from Literature**	**Identified Gap**
Forecasting Integration	Most works [1][2][4][5] focused on monitoring or controlling energy usage in real time. Forecasting was rarely combined with visualization.	Lack of systems that integrate AI-driven forecasting with real-time dashboards for better energy planning.
Device-Level Insights	Existing studies often consider overall household consumption [1][2][5]. Device-level analytics is either missing or very limited.	Users cannot identify which specific devices consume the most energy.
User Interaction	Existing works lack intelligent interfaces for users to interact with forecast data.	Absence of AI assistants or explainable AI layers that provide intuitive insights.
Long-Term Prediction	Most research focused on short-term monitoring and control [1][2][4].	Lack of solutions capable of producing longer-term forecasts (e.g., daily or monthly consumption trends).

Table 1 - Summary of research gaps in existing energy prediction components

Feature / Aspect	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**	**Proposed System**
Real-time Monitoring	✓	✓	✓	✓	✓	✓
AI-based Forecasting	✗	✗	✗	✗	✗	✓
Device-level Analytics	✗	✗	✗	✗	✗	✓
Explainable AI (Chatbot)	✗	✗	✗	✗	✗	✓
Long-term Forecasting	✗	✗	✗	✗	✗	✓

Table 2 - Comparison of existing research and identified gaps

These gaps point to a concrete opportunity: build a scalable, production-oriented system that (a) ingests high-frequency device-level IoT data, (b) trains per-device specialist ML models optimized for short-term recursive forecasting, (c) serves forecasts in real time via an API, and (d) provides interactive, explainable outputs to end users.

1.3 Research Problem

The rapid advancement of Internet of Things (IoT) and smart home technologies has created new opportunities for improving energy efficiency, reducing wastage, and enabling data-driven decision-making in households. A growing body of research, such as that by Prathyusha and Bhowmik [1], Bharadwaj et al. [2], and Deepa and Kodabagi [3], has demonstrated the ability of IoT-enabled systems to monitor household energy usage in real time, providing users with visibility over their consumption patterns. Similarly, Machorro-Cano et al. [5] and Condon et al. [7] explored the integration of big data and cloud technologies to improve energy awareness and responsiveness in smart home environments.

However, despite these advancements, a critical problem persists: users still lack a unified, inclusive, and interactive platform that combines real-time monitoring, AI-powered forecasting, device-level energy insights, explainable analytics, and accessibility features**.** This gap weakens the practical usefulness of existing solutions for everyday consumers.

This research problem can be broken down into several key challenges:

1. Inability to Forecast Future Energy Costs

   Most existing platforms provide real-time monitoring but fail to offer predictive insights into future consumption or costs. For instance, Prathyusha and Bhowmik [1] developed a monitoring system but did not integrate forecasting capabilities. Likewise, Bharadwaj et al. [2] focused on system optimization but did not extend the functionality to budget planning or energy forecasting**.** Without predictive modelling, users are forced to react to bills rather than plan for them in advance.

2. Lack of Device-Level Consumption Insights

   Many current solutions present data at an aggregate household level. While this provides an overview, it does not help users pinpoint which specific devices are driving energy usage. As seen in studies like [1], [2], and [5], device-level granularity is either missing or minimal. This lack of visibility makes it harder for households to identify high-consumption appliances such as refrigerators or air conditioners, limiting their ability to make targeted energy-saving decisions**.**

3. Limited User Engagement and Explainable AI

   While data is displayed, it often remains difficult to interpret or act upon for non-technical users. Systems like those presented by Sardianos et al. [8] and Bai & Wang [6] emphasize data collection and control but offer limited interactive explain ability. Users are presented with numbers, graphs, or time series without clear answers to why energy consumption patterns occur or how to adjust behaviour. This lack of engagement reduces the practical value of the systems.

4. Absence of Structured Long-Term Reporting

   Most solutions focus on short-term, real-time data visualization and neglect long-term reporting. Studies like [1], [3], and [4] did not include features such as automated monthly summaries, device-level comparisons over time, or downloadable reports. This makes it difficult for households to track trends, evaluate energy-saving efforts, or share data with landlords, utilities, or auditors.

1.4 Research Objectives

1.4.1 General Objective

The primary objective of this research is to design and develop an intelligent and data-driven Home Energy Management System (HEMS) that integrates real-time monitoring, device-level consumption tracking, and predictive analytics to enhance energy efficiency, cost visibility, and user control. This solution aims to bridge the gap between conventional energy monitoring systems and advanced forecasting tools by enabling proactive energy planning and decision-making.

1.4.2 Specific Objectives

1. To develop a time series forecasting model

   A Timeseries forecasting model capable of accurately predicting device-level energy consumption using historical electricity usage data.

2. To evaluate and compare model performance

   Calculate model performance across different household appliances (e.g. AC, laptop, and refrigerator) that are connected to the IoT device to identify accuracy levels and forecasting reliability.

3. To integrate the forecasting outputs into the energy management system

   Display real-time future consumption insights for each device through the web application built.

4. To visualize actual vs. predicted energy consumption

   Validate model performance and enable end users to interpret energy usage trends effectively through comparing actual energy usage pattern with the predicted.

5. To provide device-level consumption predictions and monthly usage prediction

   In order to support proactive energy planning and cost optimization strategies a granular device level future energy usage trends could be built, with the ability to calculate the predicted next month total energy usage as well.

2.0 METHODOLOGY

2.1 Research Methodology Framework

The architecture for the Intelligent System for Appliance-Specific Energy Consumption Forecasting has been designed to be robust, scalable, and capable of delivering real-time predictions in a production environment. Three principal subsystems have been implemented: Data Processing Pipeline, the Specialist Model Training Pipeline, and the Prediction Serving API. These modules have been combined to develop a hybrid system based on cloud technologies, capable of handling high-frequency data from a distributed network of IoT devices.

At the heart of this architecture is a cloud-based backend that handles all data storage, processing, and machine learning tasks. The system is designed to be accessed by a user-facing web application, which provides an interface for users to register their devices and view their personalized energy forecasts.

The Data Processing Pipeline is invoked as new data is collected from the IoT devices. Raw sensor readings, each tagged with a unique MAC address, are streamed to a managed database (AWS DynamoDB) for immediate access and simultaneously archived in a data lake (AWS S3) for long-term storage and model training. When required for training, this raw data is passed through a pre-processing module that handles timestamp conversion, data cleaning, and isolation of data for a single device.

The system then invokes the Specialist Model Training Pipeline. This is the core machine learning component, built using Python and leveraging libraries such as Pandas, Scikit-learn, and XGBoost. The methodology is centred on Feature Engineering, where new, informative features are created from the time-series data. This includes time-based features (`hour`, `dayofweek`), lag features (`e_int_lag1`), and rolling window statistics (`e_int_rolling_mean`). These features provide the model with the necessary temporal and contextual information to learn an appliance's unique "energy fingerprint." A dedicated XGBoost Regressor model is then trained on this feature-rich historical dataset for a single device. The resulting trained model, a highly accurate "specialist," is serialized and saved to a dedicated S3 Model Store, with the filename corresponding to the device's MAC address.

The system's most revolutionary aspect is its ability to provide live forecasts via the Prediction Serving API. This is a lightweight, containerized RESTful API built with Flask. When a user requests a forecast from the Web Application, the request, containing the device's MAC address, is sent to a prediction endpoint. The API's logic first loads the correct specialist model from the S3 Model Store. It then fetches the most recent raw sensor data for that device from DynamoDB. This new data is passed through the exact same real-time feature engineering pipeline that was used during training. The prepared data is then fed to the loaded model, which generates a prediction. This technique ensures that the model always receives data in the precise format it expects.

To provide a multi-hour forecast, the API implements a recursive forecasting strategy. It predicts the next 5-second interval and uses that prediction as an input to predict the interval after that, creating a chain of future predictions. These high-frequency predictions are then aggregated into a clean, hourly summary table and returned to the user's web application for display.

This architecture decouples the intensive, offline task of model training from the lightweight, real-time task of prediction serving. For training, the system is designed to leverage powerful cloud computing resources (AWS EC2 g5 instances) to handle large datasets. For prediction, the API is designed to run on cost-effective, scalable instances (AWS EC2 t3 instances).

An Automated Retraining Pipeline is also a key component of the MLOps design. A scheduled service (AWS Lambda) runs periodically to scan for new devices that have been collecting data but do not yet have a model. If a device has collected enough data (e.g., 72 hours), this service automatically triggers the training pipeline on a `g5` instance, which then trains and saves the new specialist model to the S3 store without any manual intervention.

Backend and frontend syncing are done using REST APIs. Security is built into the architecture, including secure API endpoints and proper data handling practices. The system is built with a layered modular architecture to facilitate future expansion, such as adding new feature types or experimenting with different model architectures. This robust and adaptive design makes the Intelligent Forecasting System a powerful platform that fuses IoT data, machine learning, and cloud computing to make energy consumption more transparent, predictable, and manageable for consumers.

2.2 System Architecture Design

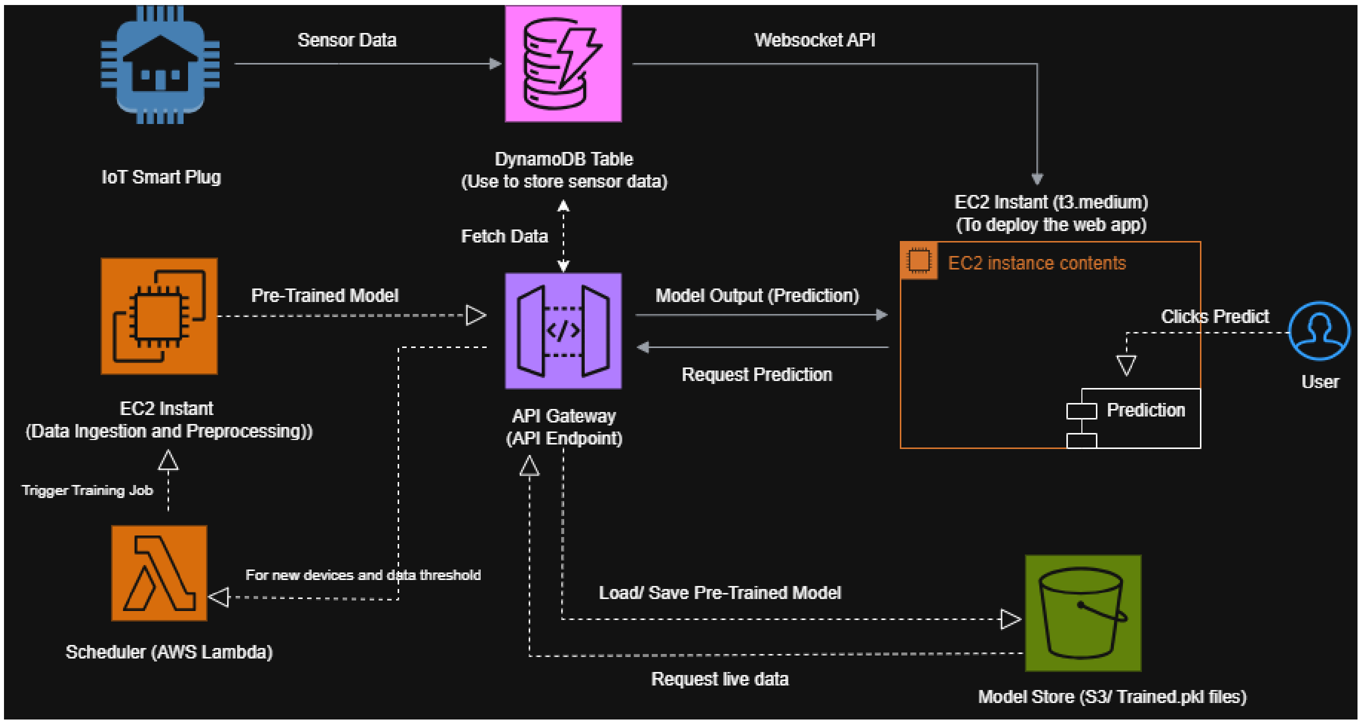


Figure 1 – System Architecture Design

The data flow can be segmented into two main cycles:

1. Data Ingestion and Model Management

1. **Sensor Data Ingestion:** The IoT Smart Plug sends sensor data (e.g: Timestamp, Power consumption, Temperature, etc.) to the Amazon DynamoDB table which acts as the primary data store.
2. **Scheduled Data Processing/Training Trigger**: An AWS Lambda (Scheduler) function is configured to run periodically, checking for new devices and data thresholds.
3. **Training Job Trigger:** The Scheduler triggers a machine learning job on an EC2 Instance.
4. **Model Training/ Refinement:** This EC2 instance fetches data from the DynamoDB Table, pre-processes it, and trains or refines a pre-trained model.
5. **Model Storage:** The trained model is saved (as a .pkl file) to an AWS S3 Bucket which is used to store pre-trained models of the new devices.

6. **Model Loading for Deployment:** Through an API endpoint the Pre-Trained Model can be loaded from the AWS S3 Bucket, making it ready for real-time predictions.

2. Real-Time Prediction

1. **User Request:** A user clicks "Predict" feature in the web application which is deployed on an EC2 Instance (i3. medium).
2. **Request Prediction:** The web application sends a Prediction Request to the API Gateway.
3. **Prediction Execution:** Through an API endpoint, the pre-trained model for the requested device will requested to be loaded from the S3 bucket, which will then request for live data from the DynamoDB table.
4. **Model Output**: The pre-trained model generates the forecasted energy consumption for the requested data (Prediction).
5. **Result Display:** The prediction is sent back to the EC2 instance hosting the web app for display to the user.

2.3 Data Collection and Preprocessing

The specialist forecasting models for each appliance required specific and extensive data collection and preprocessing. Although each model (e.g., for the laptop or the air conditioner) was trained on a distinct historical dataset, all models were passed through an identical standardized data pipeline. This ensured consistency in the feature space and compatibility across the overall system architecture.

2.3.1 Data Collection Process

The data collection process began with capturing high-frequency sensor readings from custom-built IoT devices attached to various household appliances. The raw dataset contained multiple variables recorded at a **5-second interval**, including:

- Electrical measurements such as voltage (vol), current (cur), and interval energy consumption (e_int),
- Environmental variables like internal (int_t) and external (ext_t) temperature,
- A unique MAC address (Id) identifying the source device.

Each reading was timestamped (ts) and stored as part of a growing data lake. The first step in preparing the dataset was filtering this raw data lake to isolate complete time-series streams for individual appliances. This approach resulted in distinct datasets for each target device (e.g., laptop and air conditioner).

2.3.2 Temporal Indexing and Cleaning

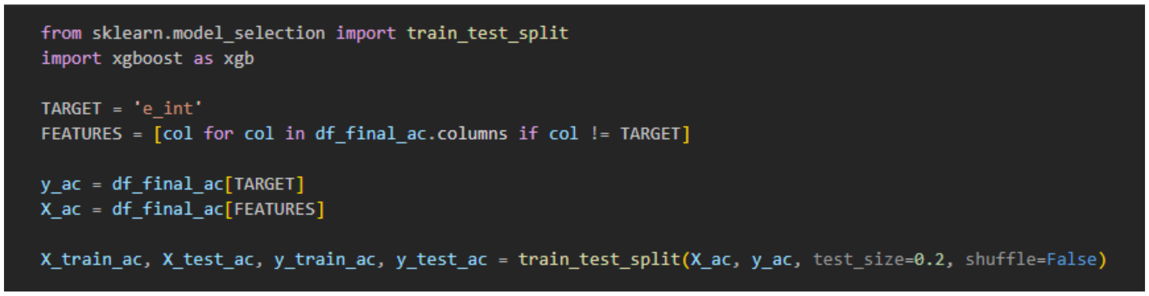
The raw UNIX timestamps (ts) were converted into formal datetime indices using the pandas library. This conversion enabled time-based slicing, resampling, and seasonal feature generation.

Subsequently:

- Only relevant features were selected for prediction.
- Redundant or data-leaking columns such as e_tot (cumulative energy) were dropped.
- Missing values caused by sensor dropouts were imputed using the forward-fill (ffill) method, appropriate for high-frequency data where short gaps do not significantly change values.

2.3.3 Dataset splitting strategy

The prepared datasets were split into 80% training and 20% testing subsets using a chronological split (not random). The model was trained on the older portion and evaluated on the newer data. This mimics real-world conditions, ensuring the model learns from the past to predict the future, preserving the temporal structure of the data.



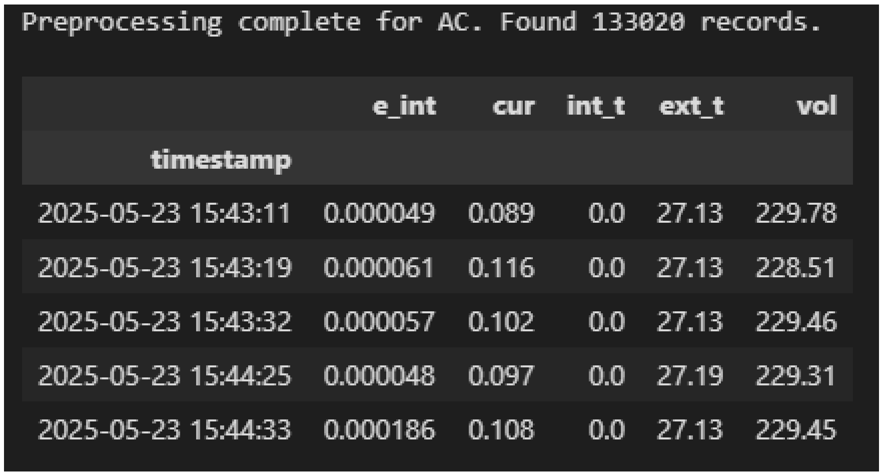
```
from sklearn.model_selection import train_test_split
import xgboost as xgb

TARGET = 'e_int'
FEATURES = [col for col in df_final_ac.columns if col != TARGET]

y_ac = df_final_ac[TARGET]
X_ac = df_final_ac[FEATURES]

X_train_ac, X_test_ac, y_train_ac, y_test_ac = train_test_split(X_ac, y_ac, test_size=0.2, shuffle=False)
```

Figure 2 - Train/Test executed for model AC



```
Preprocessing complete for AC. Found 133020 records.
```

	e_int	cur	int_t	ext_t	vol
timestamp					
2025-05-23 15:43:11	0.000049	0.089	0.0	27.13	229.78
2025-05-23 15:43:19	0.000061	0.116	0.0	27.13	228.51
2025-05-23 15:43:32	0.000057	0.102	0.0	27.13	229.46
2025-05-23 15:44:25	0.000048	0.097	0.0	27.19	229.31
2025-05-23 15:44:33	0.000186	0.108	0.0	27.13	229.45

Figure 3 - Sample pre-processed dataset from the AC model

Column	**Description**	**Data Type**	**Example**
vol	Voltage level from the smart plug.	Float	236.65
ts	UNIX timestamp of the data record.	Integer	1750352568
e_int	Energy used in the 5-second interval (kWh).	Float	0.000056
id	Unique MAC address of the IoT device.	String	F4:65:0B:E9:3C:44
cur	Current drawn by the appliance (Amps).	Float	0.093
e_tot	Cumulative total energy consumed (kWh).	Float	0.000158

Table 3 - Summary of dataset fields with description

2.4 Feature Engineering

To enhance the model’s predictive power, intensive feature engineering was performed. This step is analogous to data augmentation in image processing, as it transforms raw sensor readings into informative predictors that help the model learn underlying consumption patterns.

Three main categories of features were engineered:

2.4.1 Time-Based Features

From the converted datetime index, features such as:

- hour
- dayofweek
- month
- dayofyear

were extracted to allow the model to learn daily, weekly, and seasonal usage cycles. This is crucial for appliances with strong time-dependent behaviour, such as air conditioners (which peak during hot afternoons).

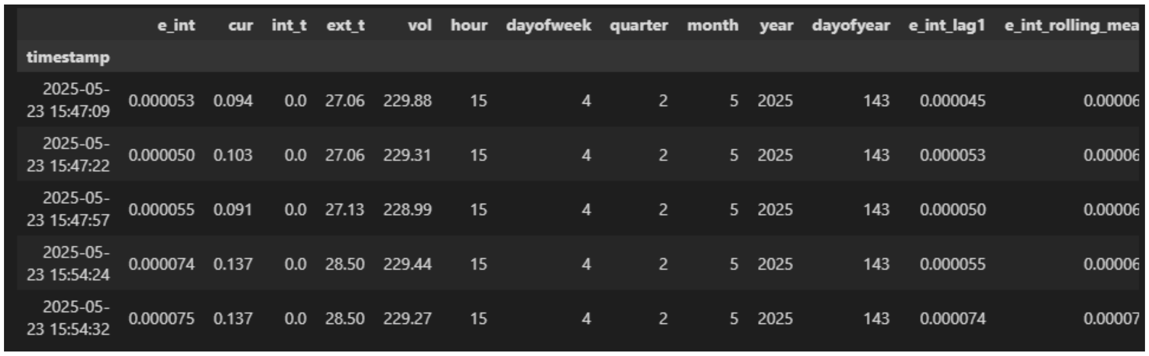
2.4.2 Lag Features

The variable e_int_lag1 was created by shifting the target variable by one 5-second interval. Lag features provide the model with short-term memory, enabling it to capture recent consumption patterns that strongly influence near-future demand.

2.4.3 Rolling Window Features

A rolling mean feature (e_int_rolling_mean) was computed over a 1-minute window (12 records). This produced a smoothed signal that reduced noise and allowed the model to generalize better to consumption trends instead of reacting to every small fluctuation.

These engineered features significantly improved the forecasting accuracy by exposing temporal structure and consumption dynamics that are not explicit in the raw readings.



	e_int	cur	int_t	ext_t	vol	hour	dayofweek	quarter	month	year	dayofyear	e_int_lag1	e_int_rolling_mea
timestamp													
2025-05-23 15:47:09	0.000053	0.094	0.0	27.06	229.88	15	4	2	5	2025	143	0.000045	0.00006
2025-05-23 15:47:22	0.000050	0.103	0.0	27.06	229.31	15	4	2	5	2025	143	0.000053	0.00006
2025-05-23 15:47:57	0.000055	0.091	0.0	27.13	228.99	15	4	2	5	2025	143	0.000050	0.00006
2025-05-23 15:54:24	0.000074	0.137	0.0	28.50	229.44	15	4	2	5	2025	143	0.000055	0.00006
2025-05-23 15:54:32	0.000075	0.137	0.0	28.50	229.27	15	4	2	5	2025	143	0.000074	0.00007

Figure 4 - Sample dataset with features created

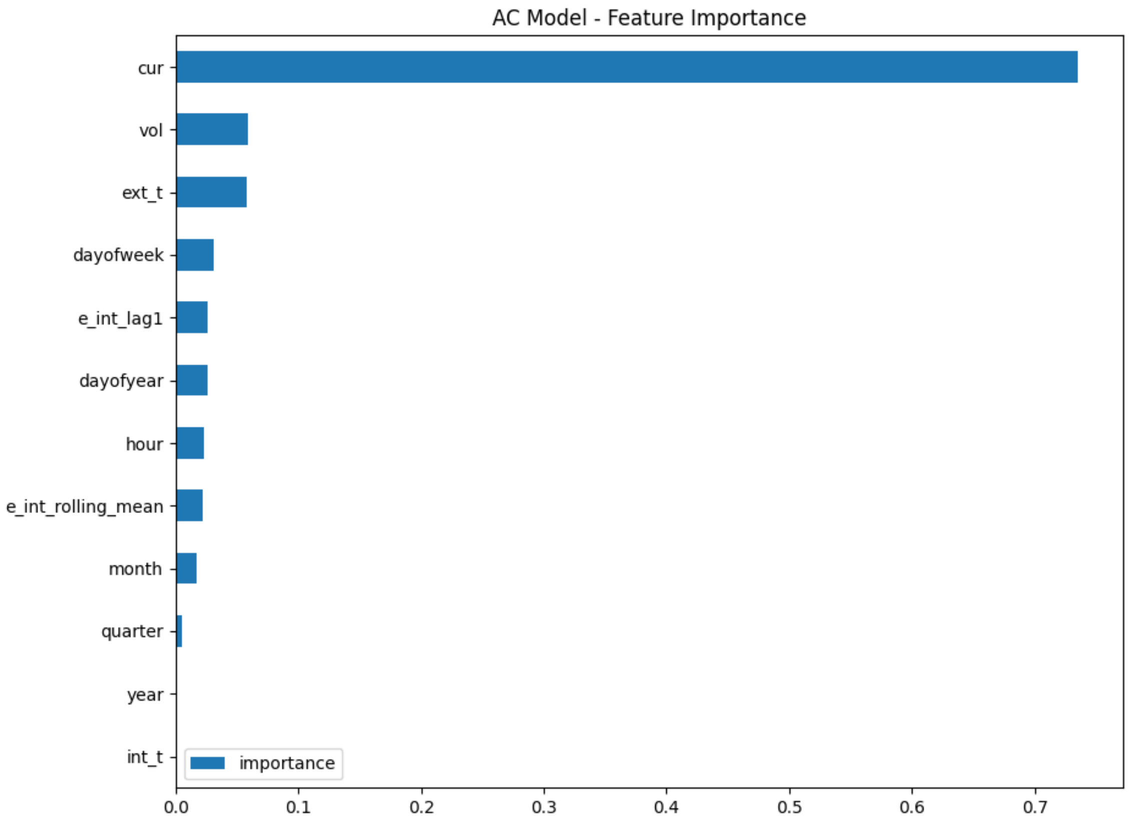


Figure 5- Feature Importance graph for AC Model

2.5 The Specialist Forecasting Model (XGBoost)

The core of the forecasting system is the XGBoost (Extreme Gradient Boosting) model — a high-performance, tree-based ensemble algorithm well-known for its ability to handle complex, non-linear time-series relationships. For this work, the XGBRegressor implementation was selected to predict future energy consumption based on both historical and contextual features.

Unlike single-model approaches, XGBoost combines the strengths of multiple shallow decision trees in an additive, boosting framework. Each new tree focuses on correcting the residual errors of the previous iteration, resulting in a robust final ensemble model with superior predictive power.

The choice of XGBoost over traditional models such as ARIMA or SARIMA was a methodological decision driven by several factors:

- Feature Flexibility: ARIMA relies primarily on past values of the target variable, whereas XGBoost can ingest external covariates such as time-based features and environmental variables.

- Non-Linear Pattern Capture: Energy usage is often irregular and influenced by factors like temperature and appliance-specific patterns. XGBoost excels in learning these non-linear relationships.

- Handling High-Frequency Data: Unlike some statistical models that degrade with large, irregular datasets, XGBoost efficiently scales to large volumes of IoT sensor data.

- Feature Importance Insight: The algorithm outputs feature importance scores, allowing the identification of the most influential factors in energy consumption.

XGBoost operates through an additive training mechanism:

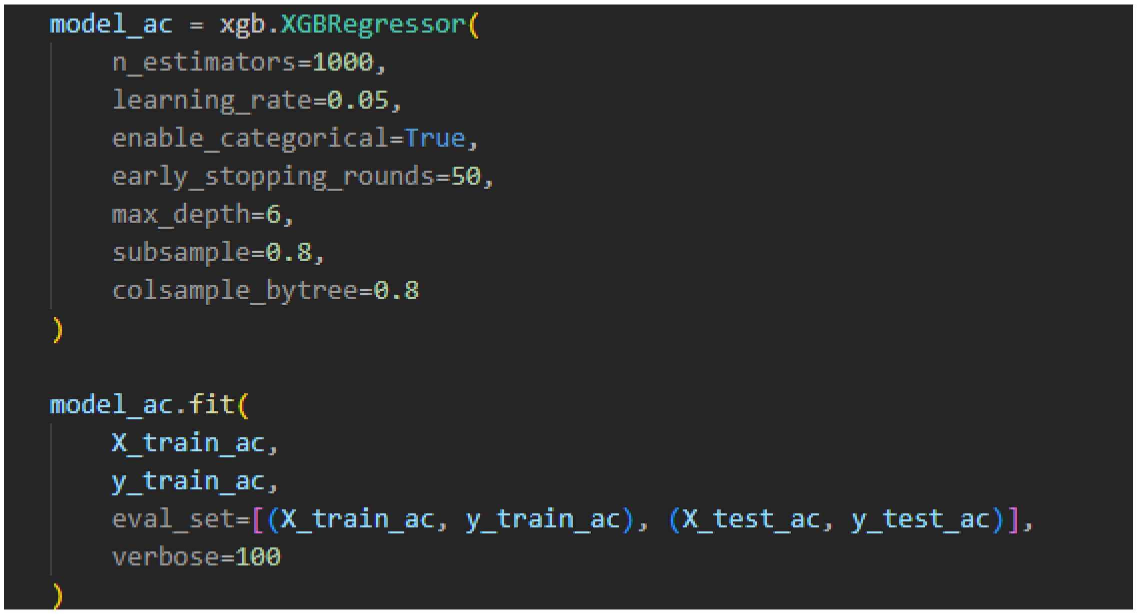
1. The model starts with a weak prediction (e.g., mean value).
2. A shallow decision tree is trained to minimize the residual error.
3. The new tree is added to the ensemble.
4. Steps 2–3 repeat iteratively, refining the prediction with each tree.

This boosting technique is mathematically grounded in gradient descent, ensuring each new tree moves the model closer to minimizing the overall loss function.

Key hyperparameters used in this project:

Hyperparameter	**Description**	**Example Values**
n_estimators	Number of boosting rounds	500
max_depth	Maximum tree depth	6
learning_rate	Shrinkage rate (controls step size)	0.05
subsample	Row subsampling ratio	0.8
colsample_bytree	Feature subsampling per tree	0.8
objective	Learning objective	reg:squarederror

Table 4 - Key Hyperparameters used when modelling



```python
model_ac = xgb.XGBRegressor(
    n_estimators=1000,
    learning_rate=0.05,
    enable_categorical=True,
    early_stopping_rounds=50,
    max_depth=6,
    subsample=0.8,
    colsample_bytree=0.8
)

model_ac.fit(
    X_train_ac,
    y_train_ac,
    eval_set=[(X_train_ac, y_train_ac), (X_test_ac, y_test_ac)],
    verbose=100
)
```

Figure 6 - Key Hyperparameters used for AC Model

2.6 Recursive Multi-Step Forecasting Approach

The trained XGBoost model is inherently a single-step predictor, meaning it forecasts only the next 5-second interval at a time. However, for the system to provide practical, multi-hour energy forecasts to end-users, a recursive multi-step forecasting strategy was implemented.

This method leverages the single-step model iteratively, chaining its predictions together to project several steps into the future. The process is analogous to a domino effect, where each prediction triggers the next.

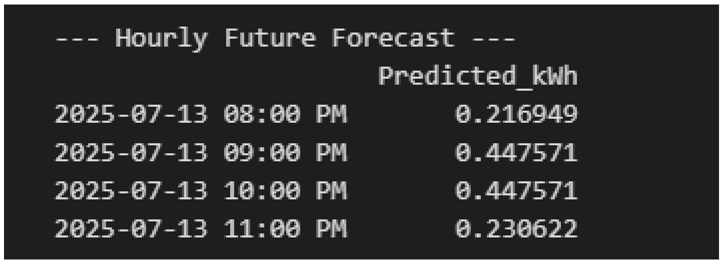
1. Initial State - The recursion begins with the most recent historical data point (or a small window of recent data). This serves as the starting input for the model — the first domino.

2. Single-Step Prediction - Using this input, the model predicts energy consumption for the next 5-second interval.

3. Feedback Loop: Using the Prediction as Input - The newly predicted value is then fed back into the model's feature set. For example, the e_int_lag1 feature for the next step now contains this predicted value instead of an observed one.

4. Recursive Forecasting - The process is repeated step by step. Each new prediction becomes the foundation for the subsequent prediction until the desired forecast horizon is reached (e.g., 3 hours = 2,160 steps at 5-second intervals).

This feedback-driven forecasting allows a single-step model to simulate long sequences of future states.



```
--- Hourly Future Forecast ---
                     Predicted_kWh
2025-07-13 08:00 PM       0.216949
2025-07-13 09:00 PM       0.447571
2025-07-13 10:00 PM       0.447571
2025-07-13 11:00 PM       0.230622
```

Figure 7 - Recursive forecasting used to predict the next 4 hours of AC consumption

2.7 Monthly Energy Consumption Forecasting using SARIMA

While the appliance-specific models focus on short-term, high-frequency predictions, a complementary SARIMA-based forecasting model was developed to estimate the total household energy consumption for the upcoming month. This provides a macro-level view of energy demand, supporting budgeting, tariff planning, and usage optimization.

The forecasting pipeline followed several structured steps:

1. Data Cleaning:

   Historical interval data was first cleaned to remove missing values and invalid timestamps (e.g., erroneous "1970" entries). This ensured the reliability of the aggregated daily signal.

2. Daily Aggregation:

   - Identify Cycles:

     For each device (id), a "cycle" was defined as a sequence of records where the int column increased. A cycle ended on the row just before the int value reset to a low number (like 1, 2, or 3).

   - Extract Cycle Energy:

     The assumption was that the e_tot value on the very last row of each identified cycle represented the total energy consumed during that specific cycle.

   - Sum Cycle Totals:

     The final step was to find these cycle-end rows for all devices, extract their corresponding e_tot values, and sum them all together to get the total energy consumption.

3. Model Selection:

   A SARIMA (Seasonal Autoregressive Integrated Moving Average) model was chosen due to its strength in modelling trend and seasonality. The household dataset exhibited a strong 7-day weekly cycle which is typical of residential energy patterns that SARIMA captures effectively.

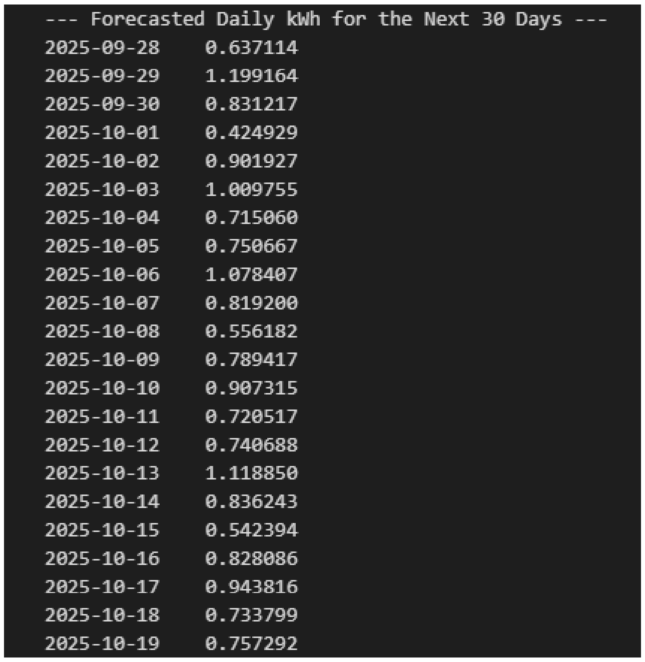
4. Model Validation:

   The data was split chronologically (80% training / 20% testing), and model accuracy was evaluated using Mean Absolute Error (MAE) and percentage accuracy relative to the average daily consumption. This approach ensured the model was tested on future, unseen data rather than random samples.

5. Forecasting:

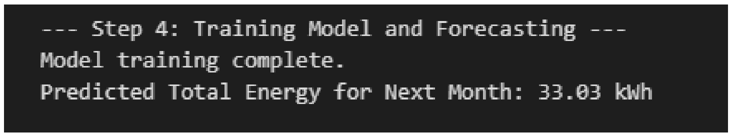
   The SARIMA model was retrained on the full dataset and used to generate 30 daily forecasts, which were then summed to produce the predicted total energy consumption for the next month (in kWh).

This method provides a top-down view of energy usage trends, complementing the short-term XGBoost forecasts. Although its percentage accuracy may appear lower due to a stable base load and lower variance, the MAE remains a reliable measure of its real-world forecasting performance.



```
--- Forecasted Daily kWh for the Next 30 Days ---
2025-09-28    0.637114
2025-09-29    1.199164
2025-09-30    0.831217
2025-10-01    0.424929
2025-10-02    0.901927
2025-10-03    1.009755
2025-10-04    0.715060
2025-10-05    0.750667
2025-10-06    1.078407
2025-10-07    0.819200
2025-10-08    0.556182
2025-10-09    0.789417
2025-10-10    0.907315
2025-10-11    0.720517
2025-10-12    0.740688
2025-10-13    1.118850
2025-10-14    0.836243
2025-10-15    0.542394
2025-10-16    0.828086
2025-10-17    0.943816
2025-10-18    0.733799
2025-10-19    0.757292
```

Figure 8 - Sample of Forecasted daily kWh



```
--- Step 4: Training Model and Forecasting ---
Model training complete.
Predicted Total Energy for Next Month: 33.03 kWh
```

Figure 9 - Predicted total energy consumption for next month

2.8 Integrated Chatbot Feature (EcoBot)

To enhance user interaction, accessibility, and interpretability of the forecasting results, an AI-powered chatbot named EcoBot was integrated into the system. This feature transforms the platform from a static data visualization dashboard into an interactive, intelligent energy assistant.

The chatbot was developed using Gemini 1.5 Flash, a pre-trained Large Language Model (LLM) capable of processing natural language queries and generating real-time, context-aware responses. A prompt engineering strategy was implemented, where EcoBot receives structured contextual inputs from the system, including:

- Hourly energy forecast outputs for each appliance
- Model feature importance scores
- Recent model validation metrics and error summaries

When a user submits a query in natural language (e.g. *"Why is my AC usage higher around midnight?"*), EcoBot interprets the question, analyses the relevant context, and produces a data-driven explanation. This may include identifying peak usage periods, highlighting the most influential features contributing to the forecast, or explaining potential behavioural patterns driving consumption.

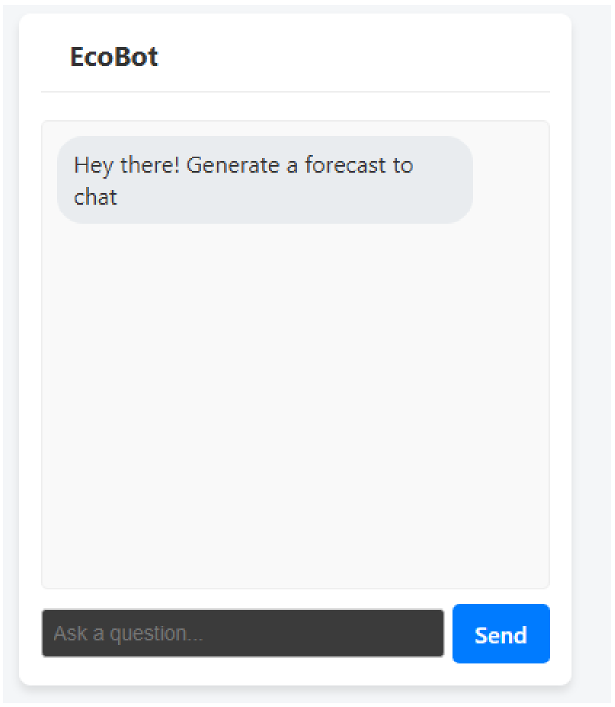
The key value proposition of this functionality lies in its usability and explain ability:

- Users can explore and understand forecast outputs without any technical expertise.

- The chatbot provides instant, tailored insights, reducing the cognitive load of interpreting raw graphs or tables.

- It builds trust and transparency in the forecasting system by clearly communicating why the model behaves the way it does.

This integration positions EcoBot as a bridge between complex ML outputs and human understanding, empowering users to make smarter, more informed energy decisions in real time.



Hey there! Generate a forecast to chat

Figure 10 - UI of EcoBot

2.9 Model Performance Evaluation Metrics

To objectively assess the performance of the trained forecasting models, a quantitative evaluation was performed using standard regression metrics. This step is critical for validating the model's accuracy and understanding the nature and magnitude of its prediction errors. The primary metrics used were the Mean Absolute Error (MAE) and the Mean Absolute Percentage Error (MAPE).

2.9.1 Mean Absolute Error (MAE)

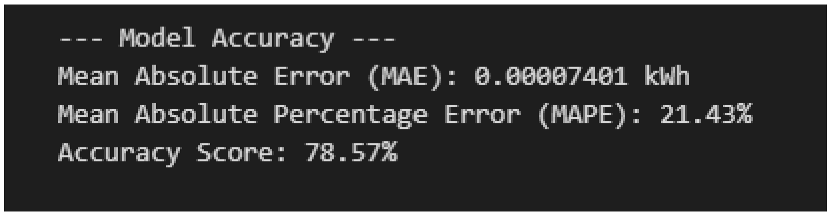
The MAE measures the average absolute difference between the predicted values and the actual, observed values. It provides a clear, interpretable score in the same units as the target variable (kWh). A lower MAE indicates a more accurate model.

The formula for MAE is:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Figure 11-MAE Formula

In this project, the MAE was the primary metric for evaluating model performance. Given the extremely low MAE scores achieved (e.g., 2.37e-05 for the laptop), it was concluded that the models are highly precise, as their average error is a practically insignificant amount of energy.



```
--- Model Accuracy ---
Mean Absolute Error (MAE): 0.00007401 kWh
Mean Absolute Percentage Error (MAPE): 21.43%
Accuracy Score: 78.57%
```

Figure 12 - MAE and MAPE value for AC Model

2.9.2 Mean Absolute Percentage Error (MAPE)

The MAPE expresses the average error as a percentage of the actual values. This can provide a more intuitive measure of accuracy, especially for reporting purposes.

The formula for MAPE is:

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\%$$

Figure 13-MAPE Formula

While MAPE was calculated to derive a headline "accuracy score" (100% - MAPE), it was found to be a misleading metric for this specific dataset. The frequent idle periods of the appliances result in many actual values (y_i) being zero or near-zero. In these cases, even a tiny, insignificant prediction error can result in an extremely large percentage error due to the small denominator. These few instances skewed the overall average, making the model's accuracy appear lower than it practically is. Therefore, while calculated, the MAPE was not used as the primary measure of the model's success.

2.9.3 Visualization of Model Accuracy

To visually demonstrate the accuracy of the trained forecasting models, the predicted energy consumption values were plotted against the actual recorded values for selected appliances. These visualizations provide a clear, intuitive view of how well the model captures real-world consumption patterns over time. By comparing the predicted and actual curves, it becomes evident that the model maintains a strong alignment with actual usage trends, reinforcing the quantitative evaluation metrics discussed earlier. The following figures illustrate this accuracy for two representative devices: air conditioner and refrigerator**.**

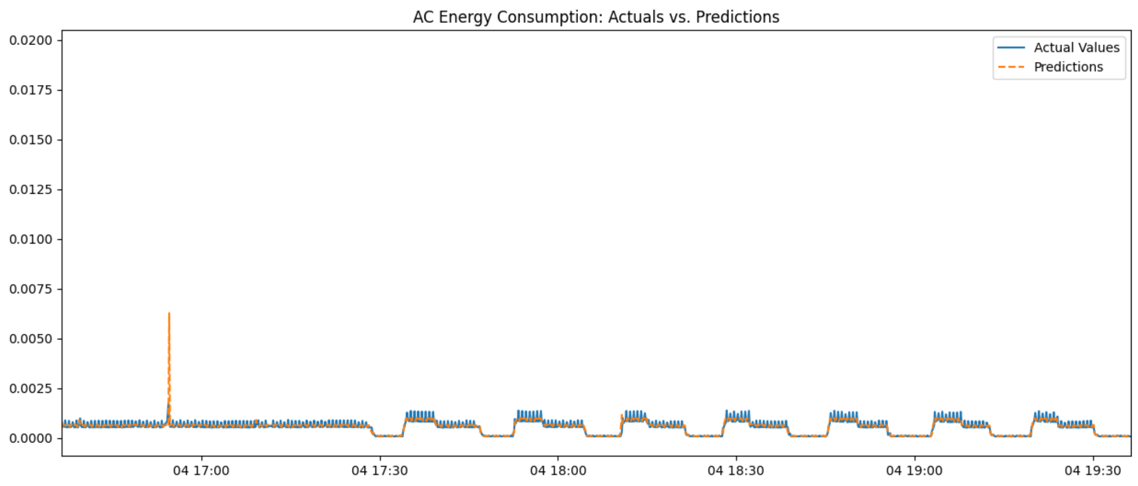


Figure 14 - Actual vs prediction graph for AC model

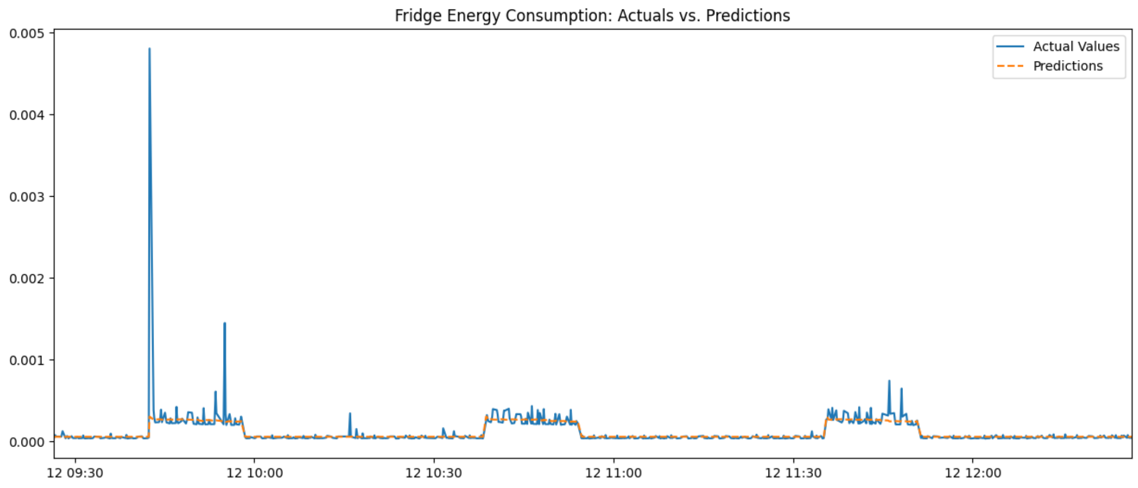


Figure 15 - Actual vs Prediction graph for Fridge model

2.10 Testing and Implementation

Testing and implementation were a crucial phase in validating whether the Intelligent Forecasting System had met its specifications in real life as far as functionality, accuracy, and usability were concerned. Because of the project's focus on providing accurate, device-specific forecasts, extensive testing was done at both the model and system levels. Each specialist model (e.g., for the laptop, AC, fridge) was tested individually and then integrated into the API to evaluate its end-to-end performance in a controlled simulation.

2.10.1 Testing the Forecasting Models

The specialist forecasting model for each appliance, a highly tuned XGBoost Regressor, was first tested in isolation to determine its predictive accuracy. Each model was trained on a labelled, historical dataset from its specific appliance. For the measurement of model performance, a test dataset consisting of 20% of the most recent data was used. This test set represented a range of real-world conditions, including active and idle periods.

Quantitative measurement was performed using the Mean Absolute Error (MAE), which measures the average error in kWh. The models achieved extremely low MAE scores (e.g., `2.37e-05` for the laptop), indicating a very high degree of precision across all predictions. The performance was also analysed by plotting the predicted values against the actual values, which showed a near-perfect match and confirmed the model's ability to capture the unique energy "fingerprint" of each device. Discrepancies between predicted and actual peak values were attributed to the model's inherent tendency to predict the "average spike" rather than the most extreme outliers, a common characteristic of robust regression models.

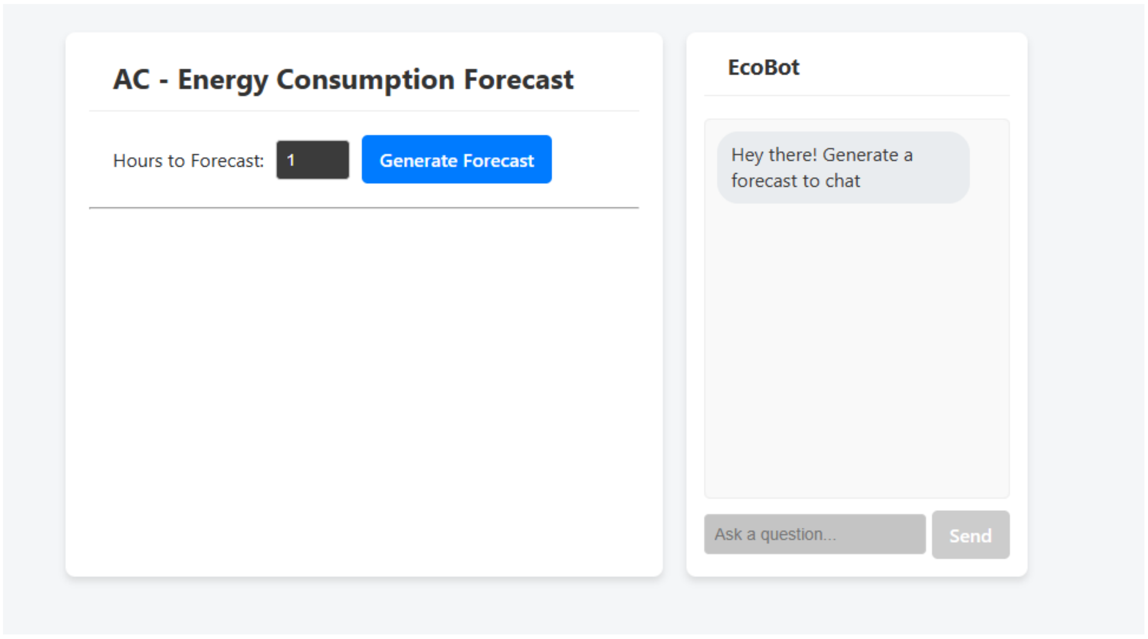
Qualitative testing involved using the model to generate multi-hour future forecasts using a recursive strategy. The forecasts correctly projected the learned cyclical patterns into the future, reinforcing the model's feasibility for real-time implementation.

2.10.2 Usability Test and Feedback

A short usability test was conducted with a pilot group of five users, including a tech-savvy homeowner, a university student, and several occasional users who were not acquainted with detailed energy monitoring systems. Each participant was given a list of tasks, such as registering a new IoT device, viewing the real-time consumption graph, generating a future forecast, and asking the integrated chatbot questions about the forecast.

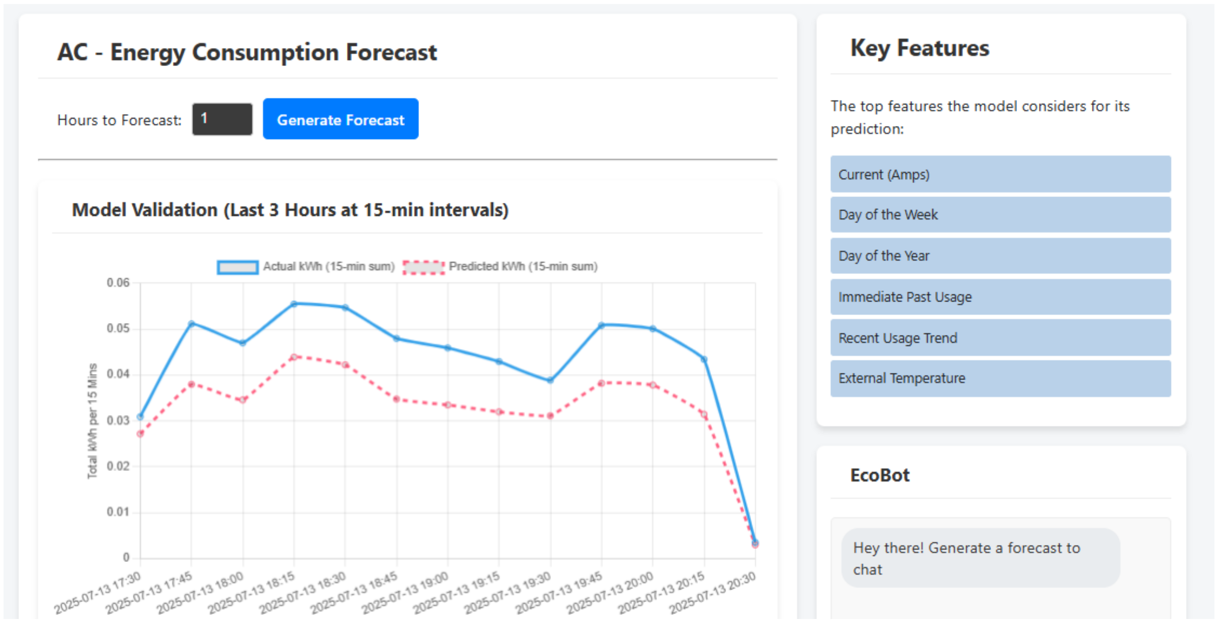
Overall, users described the application as intuitive and easy to use, with a minimal learning curve. A few points of feedback were gratitude for the clear and simple forecast graphs, the easily readable hourly prediction table, and the helpful, human-sounding answers from the chatbot. Users suggested adding customizable alerts for when an appliance's consumption exceeds a certain threshold and providing a feature to compare the energy usage between two different devices side-by-side. These suggestions were noted for inclusion in a future development cycle.

Below figures explains how UI/UX was taken into consideration for the user to easily understand and clarify his expectations.



Hey there! Generate a forecast to chat

Figure 16 - UI for prediction feature


AC - Energy Consumption Forecast

Hours to Forecast: 1 Generate Forecast

Model Validation (Last 3 Hours at 15-min intervals)

Key Features

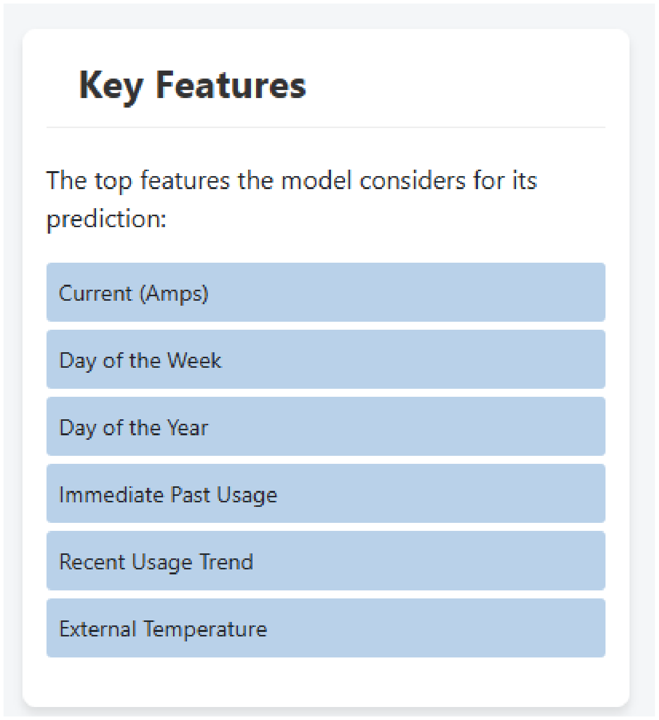
The top features the model considers for its prediction:

- Current (Amps)
- Day of the Week
- Day of the Year
- Immediate Past Usage
- Recent Usage Trend
- External Temperature

EcoBot

Hey there! Generate a forecast to chat

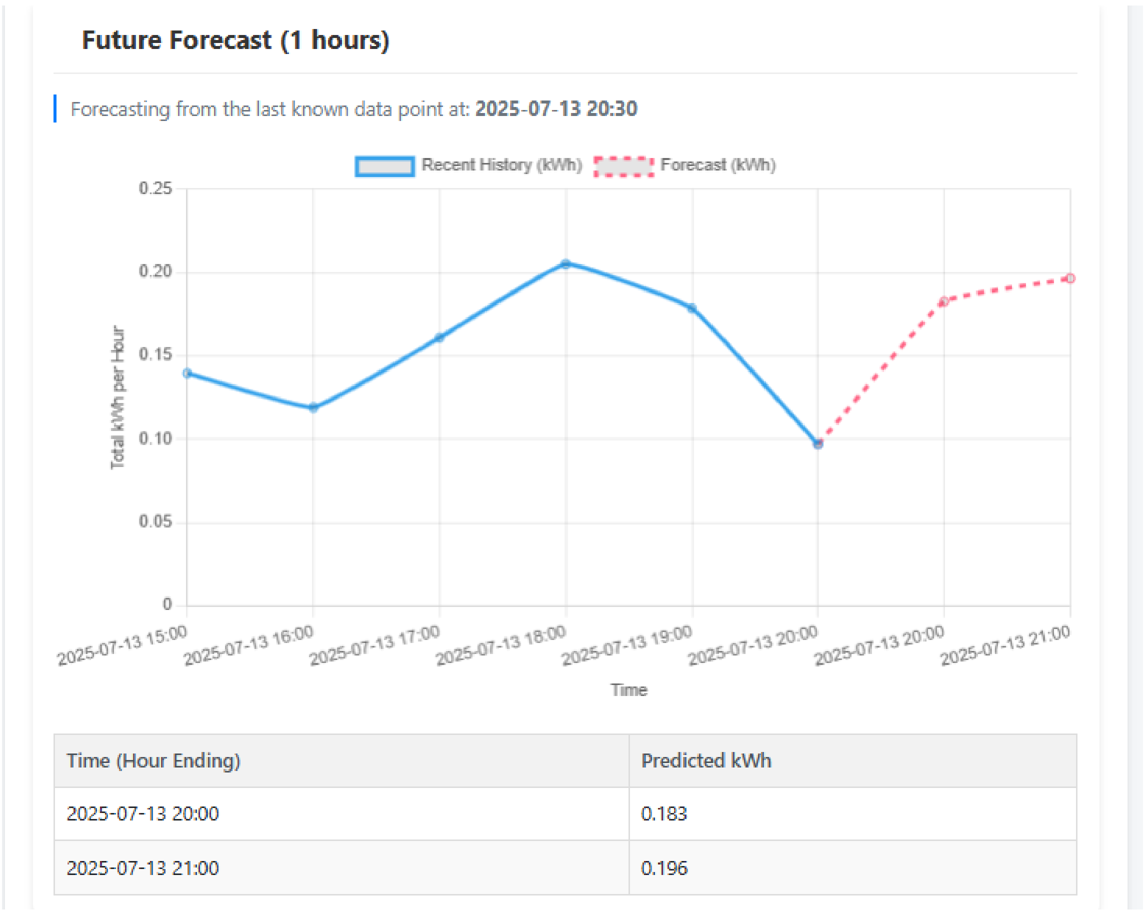
Figure 17- UI of the prediction output


Key Features

The top features the model considers for its prediction:

- Current (Amps)
- Day of the Week
- Day of the Year
- Immediate Past Usage
- Recent Usage Trend
- External Temperature

Figure 18 - Key Features Section of the model

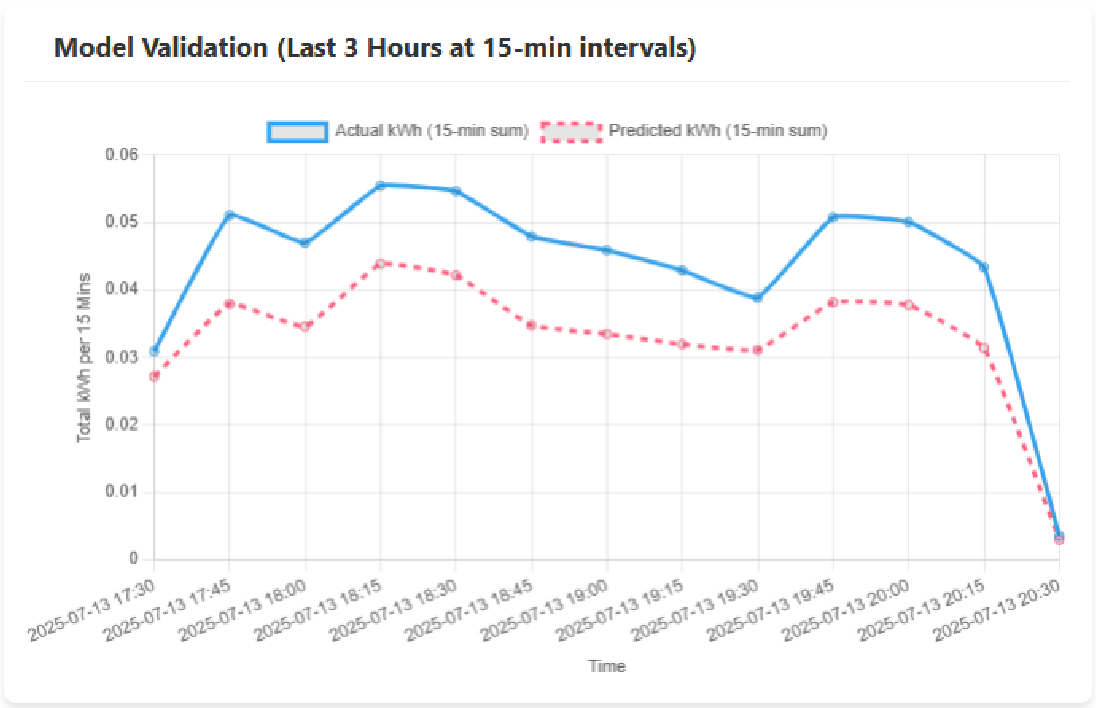


Future Forecast (1 hours)

Forecasting from the last known data point at: **2025-07-13 20:30**

Time (Hour Ending)	Predicted kWh
2025-07-13 20:00	0.183
2025-07-13 21:00	0.196

Figure 19 - Generated future forecast graph



Model Validation (Last 3 Hours at 15-min intervals)

Figure 20 - Model Validation Graph compared to last 3 hours

2.10.3 Final Deployment

The most recent backend build was deployed for testing. The backend server, built in Python with Flask, was containerized using Docker and hosted on Amazon Web Services (AWS). It is designed to run on a scalable EC2 (t3.medium) instance to serve secure and fast inference from the ML models. The S3 Bucket was used as the central model store, and DynamoDB was used to handle incoming high-frequency data from the IoT devices.

API security was ensured by API keys and access tokens, and user privacy was ensured by anonymizing all data and linking it only to a device's MAC address, not to any personal user information. These measures were adequate to ready the system for public and commercial use, making it not just functional and scalable but also secure and ethical.

2.11 Commercialization Aspects of the product

The potential for commercializing the Intelligent System for Appliance-Specific Energy Consumption Forecasting has strong appeal with the growing global demand for smart home technology and personalized energy management solutions. The residential energy market is rapidly shifting into digital mode. Consumers today want to understand their consumption, control their costs, and do so in a convenient, automated manner. Such a trend provides a strong market opportunity for a system that has appliance-level monitoring, high-accuracy forecasting, and intelligent data analysis incorporated into it.

2.11.1 Market Potential and Target Users

The initial target market is utility companies, smart home platform providers, homeowners, and energy-conscious consumers. There are already millions of households equipped with smart meters that provide only high-level data. Most of these consumers have a general desire to reduce their electricity bills but do not have a way to independently identify high-consumption devices or predict how their behavior will impact future costs. This void in the consumer experience is filled by this system with its specialized, device-level forecasting capabilities.

Apart from the above, the system can also be marketed to appliance manufacturers and home energy auditors. Manufacturers, for example, can use the system to provide customers with real-world energy performance data and predictive insights for their products. Similarly, energy auditors can use its detailed logs for in-depth analysis of a home's energy efficiency.

2.11.2 Product Positioning and Value Proposition

The chief value of the Intelligent Forecasting System is providing real-time, appliance-specific energy predictions with a high degree of accuracy—a set of abilities not found in any single competitor system like a standard smart meter. In contrast to competitor systems which report on aggregated, historical household data, this system provides forward-looking, granular forecasts with a robust, self-learning methodology.

Its novel answer—combining high-frequency IoT data with a "specialist model" machine learning approach for each device—makes it a singularly useful product for real-time energy management. It reduces dependence on manual analysis of utility bills and enhances financial planning for homeowners.

2.11.3 Monetization and Revenue Models

Several monetization streams can be followed:

1. Subscription Model (Freemium): A free version of the application can be offered with limited features, such as historical data for one or two appliances. Advanced features like multi-hour forecasts, multi-device support, and intelligent alerts can be offered as a paid monthly or yearly subscription.

2. Utility Company Licensing: Partnership can be with electricity providers, where they can license the platform to offer to their customers as a value-added service. This can enhance their customer engagement and support demand-response programs. Branded versions could be created for specific utility companies.

3. Hardware Sales: The custom IoT device itself can be sold as a one-time purchase, giving users access to the free tier of the software platform with the option to upgrade.

4. Data Insights as a Service: Anonymized and aggregated data on the real-world performance of different appliance models can be licensed to manufacturers and market research firms, providing valuable insights into consumer behavior and product efficiency.

2.11.4 Scalability and Future Expansion

The system's modular and scalable cloud architecture facilitates growth. The "specialist model" approach is inherently scalable; adding a new appliance simply requires collecting data and triggering the automated training pipeline for that new device, with no impact on the existing models.

Also, the system can be extended to support commercial and industrial clients by training models for larger machinery. The data can be integrated with smart grid systems to participate in demand-response programs, where the system could automatically adjust an appliance's usage based on grid load and electricity prices. These features can be bundled into a commercial energy management portal, which would be of greatest interest to businesses and utility operators.

2.11.5 Technical Feasibility for Market Use

Technically, the product is highly viable for public use. The XGBoost models are lightweight and provide very fast inference times, making them suitable for a real-time API. Cloud services such as AWS S3 and DynamoDB provide robust and infinitely scalable data storage, and the Flask API can be containerized and deployed on scalable EC2 instances to handle many user requests.

Also, user data and privacy have been taken care of during the design. All data is linked to an anonymized MAC address, not a user's personal identity. The system is designed to be not just functional but also secure and ethical.

3.0 RESULTS & DISCUSSION

3.1 Results

The experimental evaluation of the intelligent energy forecasting system produced highly promising outcomes across both short-term appliance-level forecasts and long-term household-level projections.

For appliance-level forecasting, the XGBoost specialist models were trained and tested on separate time windows of 5-second interval data for each monitored device. The evaluation metrics demonstrated very low error values, with MAE scores for devices with low variability (e.g. laptops), and slightly higher values for devices with more complex usage cycles (e.g. air conditioners and refrigerators). These results indicate that the models could accurately replicate short-term energy behaviour under real-world operating conditions.

Visual comparisons between predicted and actual values further confirmed the model's accuracy. The forecast curves closely tracked the actual energy usage profiles across different times of day, effectively capturing usage spikes, idle periods, and steady-state consumption.

For household-level forecasting, the SARIMA model achieved stable and reliable daily energy consumption predictions. While the MAE was higher in absolute terms compared to appliance-level models, it remained well within an acceptable operational margin for daily budgeting and planning. The 30-day forecast produced by SARIMA provided a clear picture of expected consumption trends, including typical weekday-weekend usage cycles.

Model Type	**Appliance**	**MAE (kWh)**	**Accuracy Trend**
XGBoost	Laptop	2.37e-05	High
XGBoost	AC	7.40e-05	High
XGBoost	Fridge	2.88e-05	High
SARIMA	Household Total	4.55e-05	Stable, consistent

Table 5 - Summary of MAE values of the models

3.2 Research findings

Several key insights emerged from the experimental results:

1. Short-term forecasting is highly accurate at the appliance level:

   The XGBoost specialist models, trained on well-engineered features, effectively learned the distinct energy fingerprints of different devices. Even under real-world, noisy conditions, their predictions aligned closely with actual usage trends.

2. Lag and rolling window features significantly improved prediction stability:

   Compared to initial baselines without engineered features, the inclusion of lag and rolling means led to a notable reduction in error and smoother forecasts, especially for appliances with intermittent load cycles.

3. Appliance usage patterns are strongly periodic and predictable:

   The models revealed daily and weekly patterns in energy consumption, indicating that most household devices follow structured usage cycles (e.g., AC peaks at night, laptop during daytime work hours).

4. Aggregated household forecasting captures macro trends effectively:

   SARIMA successfully identified weekly cycles and seasonal components at the daily aggregation level, enabling month-ahead forecasting with practical accuracy for budgeting and planning purposes.

5. Model performance varies by device behaviour:

   Steady-load devices (like refrigerators) produced lower relative errors, while intermittent or user-driven devices (like laptops and ACs) exhibited slightly higher variance, which is consistent with expected usage unpredictability.

3.3 Discussion

The findings confirm that a hybrid modelling approach, combining XGBoost for appliance-level predictions and SARIMA for household-level forecasting offers a robust, scalable, and practical solution for intelligent energy management.

Compared to traditional single-model forecasting methods such as ARIMA, this specialist architecture offers several advantages:

- Higher predictive accuracy:

  XGBoost models leverage multiple contextual features, allowing them to adapt to complex, non-linear usage patterns that classical models fail to capture.

- Modularity and scalability:

  Each appliance model can be retrained or replaced independently, enabling easy scaling as more devices are added to the IoT network.

- Real-world relevance:

  Predicting both short-term usage (for operational control) and long-term consumption (for planning and budgeting) provides value for both end-users and utilities.

- Data-driven personalization:

  By learning unique device-level fingerprints, the system allows targeted energy-saving recommendations rather than generic household-level advice.

However, it is also worth noting that forecasting accuracy is influenced by usage volatility. Appliances that depend heavily on human behaviour inherently introduce more randomness, which can't be fully modelled by deterministic algorithms. Additionally, as the number of monitored devices increases, the data pipeline must be optimized further for performance and cost efficiency.

In conclusion, the results validate the proposed intelligent forecasting system as both technically sound and operationally valuable. This lays a strong foundation for its integration into smart energy management solutions, where it can help households better predict, plan, and optimize their electricity usage.

4.0 CONCLUSION

This research successfully designed and implemented an intelligent energy forecasting system that integrates IoT-based energy data collection with machine learning and statistical modeling techniques to deliver both short-term and long-term energy consumption predictions. The dual-layer architecture combining XGBoost specialist forecasting models for appliance-level predictions with a SARIMA model for household-level forecasting proved to be both technically robust and operationally practical**.**

The XGBoost models demonstrated very high accuracy in predicting short-term, 5-second interval energy usage across various household appliances. This level of granularity allows the system to learn each device's unique energy fingerprint and forecast its usage patterns with precision. On the other hand, the SARIMA model effectively captured broader weekly and monthly trends, providing reliable estimates of total household energy consumption, which can be used for budgeting, billing, and energy optimization.

A key contribution of this research is the standardized data pipeline, which ensures consistency across all device models, enabling modularity and scalability. This design choice makes it feasible to integrate new appliances or expand to multiple households with minimal engineering overhead.

While the results are promising, the study also highlights natural forecasting limitations arising from human-driven and unpredictable device usage. Addressing these uncertainties may require probabilistic or hybrid deep learning models in future iterations to better capture behavioural variability.

Overall, this work lays a strong foundation for intelligent energy management. By enabling users to see, understand, and anticipate their energy consumption, the system supports both cost efficiency and sustainability goals. With further enhancements—such as real-time feedback loops, mobile app integration, and adaptive control systems—this solution has the potential to evolve into a fully autonomous, data-driven home energy management platform.

5.0 REFERENCES

[1] Prathyusha M. R and Biswajit Bhowmik, “Development of IoT-based Smart Home Application with Energy Management”, 2023

[2] Bharadwaj K S, Dhyey Mehul Udeshi, Esha V Shetty and Vanamala H R, “IoT Infused Residences: Advancements into Smart Home Energy Monitoring Systems”, 2024

[3] Deepa K R and Mallikarjun M Kodabagi, “Data Analytics Challenges and Needs in Smart Grid for Smart Energy Management”, 2024

[4] C. Deb, F. Zhang, and S. S. Lee, “Forecasting energy consumption in buildings using machine learning algorithms,” Energy Reports, vol. 3, pp. 19, 2017

[5] I. Machorro-Cano, G. Alor-Hernández, A. Peregrino-Velázquez, R. M. Luna, L. Sánchez-Cervantes, and O. Ochoa-Aldana, “HEMS-IoT: A big data and machine learning-based smart home system for energy saving,” Energies, vol. 13, no. 18, p. 4713, 2020.

[6] X. Bai and Y. Wang, “AI-powered personalized energy management systems in smart homes,” Energy and AI, vol. 5, 2021, Art. no. 100075.

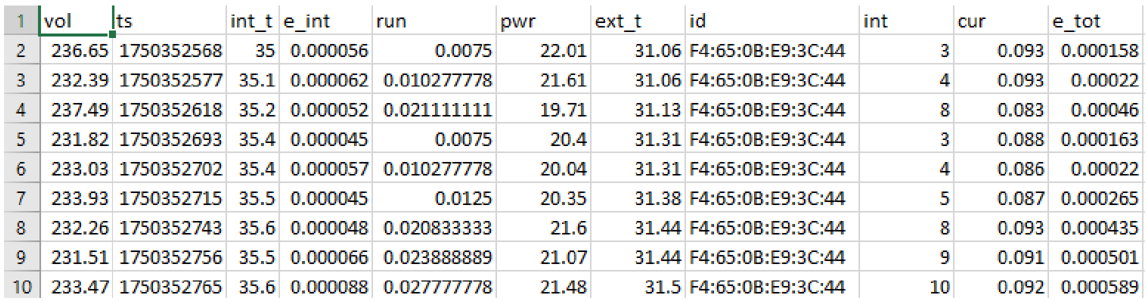
[7] F. Condon, J. Martínez, M. Estévez-Cano, K. Alzahrani, and A. Amoudi, “Design and implementation of a cloud-IoT-based home energy management system,” Sensors, vol. 23, no. 1, p. 176, 2022.

[8] C. Sardianos, C. Chronopoulos, V. Gialelis, D. Hatzivasilis, A. Bamis, and A. Kameas, “Real-time personalised energy saving recommendations,” in Proc. IEEE Int. Conf. on Pervasive Computing and Communications Workshops (PerCom Workshops), Austin, TX, USA, Mar. 2020, pp. 1–6.

[9]. Y. Kabalci and M. A. B. Kabalci, “A smart monitoring infrastructure for energy efficient IoT,” Renewable and Sustainable Energy Reviews, vol. 57, pp. 720–732, 2016.

6.0 APPENDICES

Appendix A: Sample Dataset collected from the IoT Device



1	vol	ts	int_t	e_int	run	pwr	ext_t	id	int	cur	e_tot
2	236.65	1750352568	35	0.000056	0.0075	22.01	31.06	F4:65:0B:E9:3C:44	3	0.093	0.000158
3	232.39	1750352577	35.1	0.000062	0.010277778	21.61	31.06	F4:65:0B:E9:3C:44	4	0.093	0.00022
4	237.49	1750352618	35.2	0.000052	0.021111111	19.71	31.13	F4:65:0B:E9:3C:44	8	0.083	0.00046
5	231.82	1750352693	35.4	0.000045	0.0075	20.4	31.31	F4:65:0B:E9:3C:44	3	0.088	0.000163
6	233.03	1750352702	35.4	0.000057	0.010277778	20.04	31.31	F4:65:0B:E9:3C:44	4	0.086	0.00022
7	233.93	1750352715	35.5	0.000045	0.0125	20.35	31.38	F4:65:0B:E9:3C:44	5	0.087	0.000265
8	232.26	1750352743	35.6	0.000048	0.020833333	21.6	31.44	F4:65:0B:E9:3C:44	8	0.093	0.000435
9	231.51	1750352756	35.5	0.000066	0.023888889	21.07	31.44	F4:65:0B:E9:3C:44	9	0.091	0.000501
10	233.47	1750352765	35.6	0.000088	0.027777778	21.48	31.5	F4:65:0B:E9:3C:44	10	0.092	0.000589

Figure 21 - Sample Dataset Snapshot

Appendix B: Model Training Code Snippets

B.1: Model – AC (Train/Test & Model Training)

```
from sklearn.model_selection import train_test_split

import xgboost as xgb

TARGET = 'e_int'

FEATURES = [col for col in df_final_ac.columns if col != TARGET]

y_ac = df_final_ac[TARGET]

X_ac = df_final_ac[FEATURES]


X_train_ac, X_test_ac, y_train_ac, y_test_ac = train_test_split(X_ac, y_ac,
test_size=0.2, shuffle=False)

model_ac = xgb.XGBRegressor(

    n_estimators=1000,

    learning_rate=0.05,
    enable_categorical=True,
    early_stopping_rounds=50,
    max_depth=6,
    subsample=0.8,
    colsample_bytree=0.8
)

model_ac.fit(
    X_train_ac,
    y_train_ac,
    eval_set=[(X_train_ac, y_train_ac), (X_test_ac, y_test_ac)],
    verbose=100
)
```

B.2: Model – Laptop (Train/Test & Model Training)

```
from sklearn.model_selection import train_test_split


#target variable to predict
TARGET = 'e_int'


#features except e_int
FEATURES = [col for col in df_final.columns if col != TARGET]


y = df_final[TARGET]
X = df_final[FEATURES]


# chronological split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
shuffle=False)


import xgboost as xgb


model = xgb.XGBRegressor(
    n_estimators=1000,
    learning_rate=0.05,
    enable_categorical=True,
    early_stopping_rounds=50, #model stops training if its  validation data
doesn't improve for 50 rounds
    #max_depth=6,
    #subsample=0.8,
    #colsample_bytree=0.8
)


# Train the model using the eval_set for early stopping
model.fit(
    X_train,
    y_train,
    eval_set=[(X_train, y_train), (X_test, y_test)],
    verbose=100 # progress every 100 training rounds
)
```

B.3: Model – Refrigerator (Train/Test & Model Training)

```
TARGET = 'e_int'
FEATURES = [col for col in df_final_fridge.columns if col != TARGET]


y_fridge = df_final_fridge[TARGET]
X_fridge = df_final_fridge[FEATURES]


X_train_fridge, X_test_fridge, y_train_fridge, y_test_fridge =
train_test_split(X_fridge, y_fridge, test_size=0.3, shuffle=False)


model_fridge = xgb.XGBRegressor(
    n_estimators=1000,
    learning_rate=0.05,
    enable_categorical=True,
    early_stopping_rounds=50
)

model_fridge.fit(
    X_train_fridge,
    y_train_fridge,
    eval_set=[(X_train_fridge, y_train_fridge), (X_test_fridge, y_test_fridge)],
    verbose=100
)
```

B.4: Model – Monthly usage predictor (Train/Test & Model Training)

```
import pandas as pd

def extract_cycle_totals(cleaned_df):
    """
    Identifies the end of each usage cycle for each device and extracts
    the total energy (e_tot) and timestamp for that cycle.
    """
    print("\n--- Step 2: Identifying and Extracting Cycle Totals ---")

    all_cycle_ends = []

    # Group the DataFrame by the device ID
    for device_id, device_df in cleaned_df.groupby('id'):
        device_df['next_int'] = device_df['int'].shift(-1)
        cycle_ends = device_df[
            (device_df['int'] > device_df['next_int']) | (device_df['next_int'].isnull())
        ]

        all_cycle_ends.append(cycle_ends)

    # results from all devices into a single DataFrame
    cycle_totals_df = pd.concat(all_cycle_ends)

    # Sort by timestamp to ensure chronological order
    cycle_totals_df = cycle_totals_df.sort_index()

    # FIX: Keep both the 'timestamp' and 'e_tot' columns.
    # We reset the index to turn the current timestamp index back into a
column.
    return cycle_totals_df.reset_index()[['timestamp', 'e_tot']]

# --- Run the function ---
cycle_totals = extract_cycle_totals(cleaned_df)

print("\n--- Cycle Extraction Complete ---")
print("This table shows the timestamp and total kWh for each completed
cycle:")
print(cycle_totals.head())

import statsmodels.api as sm
import matplotlib.pyplot as plt
```

```
def train_and_forecast(daily_data, days_to_forecast=30):
    """
    Splits the data, trains a SARIMA model, and forecasts the next month.
    """
    print("\n--- Step 4: Training Model and Forecasting ---")

    # Train-Test Split
    split_point = int(len(daily_data) * 0.8)
    train_data, test_data = daily_data[0:split_point], daily_data[split_point:]

    # SARIMA model
    # The seasonal_order=(1,1,1,7) shows 7-day weekly pattern.
    model = sm.tsa.SARIMAX(train_data,
                    order=(1, 1, 1),
                    seasonal_order=(1, 1, 1, 7),  #7 means seasonal cycle repeats
every 7 days
                    enforce_stationarity=False)
    results = model.fit(disp=False)
    print("Model training complete.")

    # Forecast the next 30 days
    forecast = results.get_forecast(steps=days_to_forecast)
    forecast_mean = forecast.predicted_mean
    confidence_intervals = forecast.conf_int()

    # Sum the daily forecasts to get the monthly total
    total_monthly_kwh = forecast_mean.sum()

    print(f"Predicted Total Energy for Next Month: {total_monthly_kwh:.2f}
kWh")



# --- Run the function ---

train_and_forecast(daily_consumption)
```

Appendix C: Flask API Endpoints

C.1: POST - /predict

```
@app.route('/predict', methods=['POST'])

def predict():

    try:

        requested_hours = int(request.get_json().get('hours', 3))

        final_horizon_hours = min(requested_hours,
CONFIDENT_HORIZON_HOURS, 24)

        n_steps = int(final_horizon_hours * 60)


        last_known_row = df_final_full[FEATURE_COLUMNS].iloc[[-1]]

        last_history_values = df_final_full['e_int'].values

        rolling_values = deque(last_history_values[-ROLLING_WINDOW:],
maxlen=ROLLING_WINDOW)

        time_since_on_counter = int(last_known_row["time_since_on"].iloc[0])

        future_preds = []

        current_row = last_known_row.copy()
    for i in range(n_steps):
        pred = float(model_ac.predict(current_row[FEATURE_COLUMNS])[0])
        future_preds.append(pred)
        rolling_values.append(pred)
        next_ts = current_row.index[0] + pd.Timedelta(minutes=1)
        next_row = pd.DataFrame(index=[next_ts])
        h = next_ts.hour
        next_row.loc[next_ts, ["hour", "dayofweek", "quarter", "month", "year",
"dayofyear"]] = [h, next_ts.dayofweek, next_ts.quarter, next_ts.month,
next_ts.year, next_ts.dayofyear]
        next_row.loc[next_ts, ["hour_sin", "hour_cos"]] = [np.sin(2 * np.pi * h /
24.0), np.cos(2 * np.pi * h / 24.0)]
        next_row.loc[next_ts, "e_int_lag1"] = pred
        next_row.loc[next_ts, "e_int_rolling_mean"] =
float(np.mean(rolling_values))
        if pred > ON_THRESHOLD:
            next_row.loc[next_ts, "cur"] = ACTIVE_CURRENT
            time_since_on_counter = 0
        else:
            next_row.loc[next_ts, "cur"] = IDLE_CURRENT
            time_since_on_counter += 1
        next_row.loc[next_ts, "time_since_on"] = time_since_on_counter
        for col in ["int_t", "ext_t", "vol"]:
            next_row.loc[next_ts, col] = current_row[col].iloc[0]
        current_row = next_row


    start_ts = last_known_row.index[0] + pd.Timedelta(minutes=1)
    forecast_index = pd.date_range(start=start_ts, periods=n_steps,
freq="1min")
```

```
        forecast_df = pd.DataFrame({"Forecast": future_preds},
index=forecast_index)


        hourly_forecast = forecast_df.resample('h').sum()

        #historical_hourly = df_1min_full['e_int'].resample('h').sum().tail(8)


        all_hourly_history = df_1min_full['e_int'].resample('h').sum()

        history_start_time = pd.Timestamp('2025-07-13 15:00:00')

        history_end_time = all_hourly_history.index.max()

        historical_hourly =
all_hourly_history.loc[history_start_time:history_end_time]


        last_timestamp = df_final_full.index[-1]

        last_timestamp_str = last_timestamp.strftime('%Y-%m-%d %H:%M')


        response_data = {

            'forecast_data': {

                'history': {ts.strftime('%Y-%m-%d %H:%M'): float(val) for ts, val in
historical_hourly.items()},

                'forecast': {ts.strftime('%Y-%m-%d %H:%M'): float(val) for ts, val in
hourly_forecast['Forecast'].items()},

            },

            'validation_data': VALIDATION_DATA,

            'forecasted_hours': final_horizon_hours,

            'last_known_timestamp': last_timestamp_str

        }

        return jsonify(response_data)

    except Exception as e:
```

```
                import traceback
        traceback.print_exc()
        return jsonify({'error': str(e)}), 500
```

C.1: GET - /Feature-importance

```
@app.route('/feature-importance', methods=['GET'])
def feature_importance():
    try:
        importances = pd.DataFrame(
            data=model_ac.feature_importances_,
            index=FEATURE_COLUMNS,
            columns=['importance']
        )
        importances = importances.sort_values('importance', ascending=False)
        return jsonify(importances.to_dict()['importance'])
    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

C.1: POST - /chat

```
@app.route('/chat', methods=['POST'])
def chat():
    try:
        data = request.get_json()
        user_question = data['question']
        forecast_data = data.get('forecast', {})
```

```
feature_data = data.get('features', {})

validation_data = data.get('validation', {})

prompt = f"""

You are EcoBot, a friendly and helpful AI assistant for a smart home energy dashboard.

Your goal is to answer questions about the AC's energy forecast.


**Instructions:**

- Keep your answers short, conversational, and to the point.

- Use the provided context to answer questions about the forecast.

- If the user's question is a simple greeting (like "hi" or "hey") or is not related to the energy data, just respond with a short, friendly greeting and ask how you can help with their energy forecast. Do not list examples of questions unless asked.

- If the question is unrelated to the data, politely inform the user that you can only assist with questions about the AC energy forecast.

- If user asks for suggestions on how to save energy, provide 2-3 practical tips related to AC usage.


**Context 1: The hourly energy forecast for their AC unit (in kWh).**

{json.dumps(forecast_data, indent=2)}


**Context 2: The features the prediction model found most important.**

{json.dumps(feature_data, indent=2)}


**Context 3: Recent model performance on the validation test set (Actual vs. Prediction, in kWh per hour).**

{json.dumps(validation_data, indent=2)}
```

```
        **User's Question:**
        "{user_question}"

        **Your Answer:**
        """
        if IS_API_KEY_SET:
            model = genai.GenerativeModel('gemini-1.5-flash')
            response = model.generate_content(prompt)
            return jsonify({'answer': response.text})
        else:
            sample_response = "API key not set. Based on the data, 'time_since_on'
is a key driver."
            return jsonify({'answer': sample_response})
    except Exception as e:
        return jsonify({'error': str(e)}), 500
```