International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

AI-Powered Energy Consumption Optimizer

Anusha Hukkeri[1], Kanishka Gurav[2], Nagaveni Desai[3], Nandini Sarala[4], Prof. Jyoti Chitti[5]

[1,2,3,4]Student, Department of Computer Science and Engineering,

KLS Vishwanathrao Deshpande Institute of Technology, Haliyal, Karnataka, India
[5]Professor, Department of Computer Science and Engineering,
KLS Vishwanathrao Deshpande Institute of Technology, Haliyal, Karnataka, India

Affiliated to Visvesvaraya Technological University, Belagavi, Karnataka, India
[1] 2vd23cs004@klsvdit.edu.in | [2] 2vd23cs018@klsvdit.edu.in
[3] 2vd23cs031@klsvdit.edu.in | [4] 2vd23cs034@klsvdit.edu.in
[5] jsc@klsvdit.edu.in

Abstract – Households usually learn how much electricity they used only when the monthly bill arrives, by which time wasteful usage can no longer be corrected. This paper presents an AI-powered energy consumption optimizer, a web-based platform that tracks appliance-level usage, predicts the monthly bill under the Karnataka Gruha Jyothi tariff rules, and enforces a daily energy budget. Four features distinguish the work: 1) an AI Budget Guard that derives a daily allowance from the same month of the previous year, reserves energy for essential appliances, and divides the rest among the other appliances by priority-weighted shares; 2) a reward wallet that grants one point for every 10 kWh saved against the previous year, each point reducing the payable bill by ₹1; 3) a Random Forest recommendation model trained on 11,026 synthetic records derived from three monitored homes, which reached 88.40% test accuracy on three action classes; and 4) EnergyBot, a retrieval-augmented chatbot built on Hugging Face embeddings, MongoDB Atlas Vector Search, and a Groq-hosted language model. The system is implemented with React, Flask, and MongoDB, uses OpenWeatherMap data, and demonstrates appliance control through a simulated ESP32 controller in Wokwi.

Key Words: **Energy management, Bill prediction, Random Forest, Retrieval-augmented generation, Reward wallet, Smart home.**

I. INTRODUCTION

Household electricity use keeps growing as air conditioners, water heaters, refrigerators, and washing machines become standard equipment. Conventional meters report only cumulative readings: they neither attribute consumption to appliances nor forecast the bill, and they do not explain how slab rates, fixed charges, and subsidies combine into the final amount. Artificial intelligence (AI) and the Internet of Things (IoT) have been applied to smart-home energy optimization [1], a survey of AI techniques for power optimization catalogues the methods now available [2], and recent work couples load forecasting with demand-response strategies [3].

This paper presents an AI-powered energy consumption optimizer for Indian households: a web-based platform that tracks appliance-level usage, predicts the monthly bill under the Karnataka Gruha Jyothi tariff rules, and enforces a daily energy budget. The main contributions are:

1) a calendar-month bill-prediction pipeline that models the three-tier Gruha Jyothi rule (free entitlement, partial billing, and subsidy forfeiture above 200 units);
2) an AI Budget Guard that converts last year's consumption for the same month into a daily allowance, distributes it by appliance priority, and enforces it in real time, together with a virtual breaker that sheds load near the sanctioned limit;
3) a persistent reward wallet that awards one point per 10 kWh saved against the same month of the previous year;
4) a Random Forest model that recommends No_Action, Suggest_OFF, or Delay_Load for each appliance from weather, load, and priority features; and
5) EnergyBot, a retrieval-augmented generation (RAG) chatbot, and a simulated ESP32 controller that links software decisions to appliance outputs.

The remainder of this paper is organized as follows. Section II reviews related work, Section III describes the methodology, Section IV presents the results, Section V discusses their implications and limitations, and Section VI concludes the paper.

II. LITERATURE REVIEW

A. Related Work

Raj et al. [1] present an AI-driven framework for IoT-enabled smart homes in which sensors track the usage of individual appliances and machine-learning techniques analyze the data to improve energy efficiency while preserving user comfort. Biswas et al. [2] survey 17 research methodologies for AI-based power optimization, discuss their strengths and limitations, and report that decision trees and random forests consume little energy during training. This observation supports our choice of tree-based models for a small web backend.

Durrani et al. [3] compare linear regression, random forest regression, support vector regression, k-nearest neighbors, and LSTM networks on the UCI Smart Home Energy dataset under seven demand-response strategies. Their LSTM model achieved the best accuracy (MAE of 18.95 Wh, RMSE of 24.83 Wh, and R^2 of 0.94 under price-based control), while random forest was the best traditional model (MAE of 25.13 Wh and R^2 of 0.89). Whereas [3] forecasts household load, our model classifies the action to take on each appliance, and the bill is derived separately from a configurable regional tariff.

B. Research Gap

These studies emphasize forecasting, control, or infrastructure. Among the works reviewed here, none combines a regional tariff with a subsidy threshold, priority-based energy budgeting, bill-reducing rewards, and a conversational interface to the household's own data. The system described in this paper addresses this gap.

III. METHODOLOGY

A. System Overview

The platform has a presentation layer, an application layer, a data layer, a retrieval layer for the chatbot, and a simulated IoT layer, as shown in Fig. 1. The presentation layer is a React [4] single-page application that shows the dashboard, appliance manager, recommendations, settings, and chatbot. The application layer is a Flask [5] REST service that handles requests, computes energy and bills, runs the budget and reward logic, and protects routes with JSON Web Tokens. MongoDB Atlas stores users, appliances, energy logs, notifications, and the reward wallet, and its Vector Search [6] indexes the chatbot's knowledge base. Text embeddings come from the Hugging Face Inference API [7], and answers are generated by a language model hosted on Groq [8]. OpenWeatherMap supplies the temperature and humidity of the user's city. A simulated ESP32 [9] controller runs in Wokwi [10].

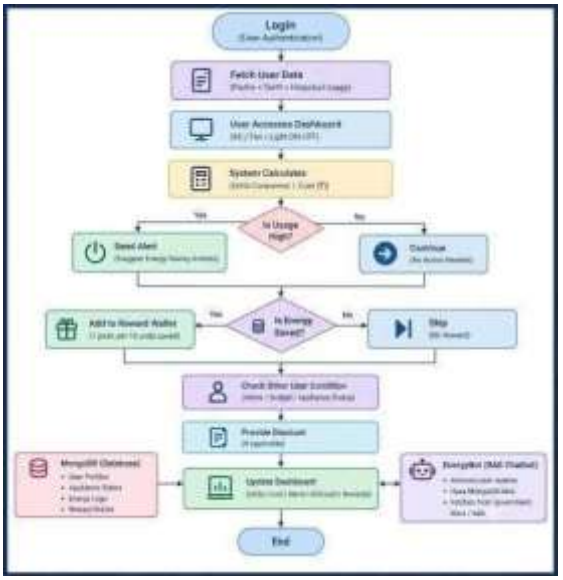


Fig. 1. Workflow of the proposed system.

B. Functional Modules

The platform comprises user authentication with SMTP-based password recovery; an appliance manager (power rating, quantity, priority class of Essential, Medium, or Non-essential, ON/OFF state, and AI timers); an energy and bill engine; a recommendation and alert engine; the AI Budget Guard; the reward wallet; and EnergyBot. Users set their historical average consumption, sanctioned load, and tariff rate in a settings page.

C. Energy Computation

For appliance i with power rating P_i (W), quantity Q_i, and daily usage h_i (h), the daily energy in kWh is

$$E_i = \frac{P_i Q_i h_i}{1000} \quad (1)$$

During operation, a simulation loop accumulates energy every second from the appliances that are ON. The monthly consumption is projected from the average daily consumption $\bar{e}$ of the days logged so far and the number of days D_m in the calendar month:

$$\hat{E}_m = \bar{e} \times D_m \quad (2)$$

D. Tariff and Bill Prediction

The tariff follows the Karnataka Gruha Jyothi structure. Let H be the household's historical average monthly consumption. The free entitlement is H increased by a 10% buffer and capped at 200 units:

$$A = \min(200, 1.10H) \quad (3)$$

For example, H = 113.53 units gives A = 124.88 units, the value shown on the dashboard in Fig. 3. The billable units u_b then follow three tiers: consumption up to A is free; consumption between A and 200 units is billed only on the excess; and above 200 units the subsidy is forfeited and every unit is billable:

$$u_b = \begin{cases} \max(0, \hat{E}_m - A), & \hat{E}_m \le 200 \\ \hat{E}_m, & \hat{E}_m > 200 \end{cases} \quad (4)$$

With the energy rate r = ₹5.80/kWh and a fixed charge of ₹150 per kW of sanctioned load S, the predicted bill is

$$B = ru_b + 150S \quad (B = 0 \text{ when } u_b = 0) \quad (5)$$

The 200-unit reference is an awareness indicator only and does not determine eligibility for the scheme. Tariff values are stored per user, so a revision requires updating a setting rather than changing code.

E. AI Budget Guard

The Budget Guard takes the consumption U_{prev} of the same calendar month of the previous year, read from a historical CSV file, as the monthly target and derives a daily house budget:

$$\beta = \frac{U_{\text{prev}}}{30} \quad (6)$$

Essential appliances (for example, the refrigerator) are assumed to run for 24 h and are reserved first. They receive no daily limit:

$$R_e = \sum_{i \in \text{Essential}} \frac{P_i \times 24}{1000} \quad (7)$$

The remaining budget $\beta - R_e$ is shared among Medium and Non-essential appliances in proportion to priority-weighted shares, where Medium appliances count twice:

$$w_i = m_i P_i, \; m_i = 2 \text{ (Medium)}, 1 \text{ (Non-essential)}$$

$$L_i = \max\left(1.0, \frac{w_i(\beta - R_e)}{\sum_j w_j}\right) \quad (8)$$

The floor of 1.0 kWh is a minimum grace allowance that keeps the household from being left without any usable budget when the arithmetic result is negative. When an appliance's accumulated energy reaches L_i, it is switched OFF and locked, and an alert is shown on the dashboard. The allocation is deterministic, rule-based logic.

Separately, a virtual breaker monitors the active load: when it exceeds 90% of the sanctioned load, Non-essential and then Medium appliances are switched off and a recommendation to wait is sent, without locking them. A heartbeat watchdog treats a 30 s silence as a power cut and blanks the dashboard.

F. Reward Wallet

Let $\hat{E}_m$ be the projected consumption of the current month. One reward point is granted for every complete 10 kWh saved relative to the same month of the previous year, and no point is granted when consumption is not lower:

$$N = \left\lfloor \frac{\max(0, U_{\text{prev}} - \hat{E}_m)}{10} \right\rfloor \quad (9)$$

Each point is worth ₹1 in the project's own reward rule. With wallet balance W, the discount applied to the predicted bill and the final bill are

$$B_f = B - \min(W, B) \quad (10)$$

Comparing the same month across years reduces seasonal effects but does not remove the influence of weather, occupancy, or appliance changes. The reward is a feature of this prototype and is not a utility discount. The wallet balance is persisted in MongoDB.

G. Recommendation Model

The model learns which action to recommend for an appliance: No_Action, Suggest_OFF (temperature- or comfort-based), or Delay_Load (overload-based). The dataset contains 11,026 records synthesized from monitoring three homes, covering 14 appliance types, with class counts of 8,727 No_Action (79.15%), 1,557 Delay_Load (14.12%), and 742 Suggest_OFF (6.73%). The labels follow the project's threshold-based recommendation rules. The inputs are the temperature and humidity, the simultaneous load, the sanctioned load, the usage hours, and an engineered load ratio,

$$\rho = \frac{1000\, L_{\text{sim}}}{S_W} \quad (11)$$

where L_{sim} is the total active load in kW and S_W is the sanctioned load in watts, plus one-hot encoded appliance name, priority, time slot, and season, giving 30 features. The data were split 80/20 with stratification (8,820 training and 2,206 test records, random seed 42). A decision tree (maximum depth 8, minimum leaf size 10) and a Random Forest of 150 trees (maximum depth 10, minimum leaf size 5) were trained. The Random Forest is the model loaded by the Flask backend; at run time its temperature and humidity inputs come from OpenWeatherMap.

H. EnergyBot

EnergyBot follows the RAG workflow shown in Fig. 2. In a one-time ingestion step, reference documents are split into segments, embedded with a Hugging Face model [7], and stored in MongoDB Atlas [6]. The knowledge base contains tariff and FAQ texts for the Karnataka distribution companies and notes on the Gruha Jyothi and Yuva Nidhi schemes. At query time, the question is embedded and the top three segments are retrieved by vector search. The retrieved text, together with the logged-in user's recent energy logs and appliance data, is sent to a Groq-hosted language model [8]. When the indexed documents lack the answer, the chatbot falls back to a web search. Account data are retrieved only for the authenticated user.



Fig. 2. EnergyBot RAG workflow.

EnergyBot provides users with personalized responses based on their electricity consumption patterns and appliance usage. It helps users understand tariff rates, available government schemes, and energy-saving methods. The chatbot can also answer queries related to electricity bills, consumption limits, and subsidy benefits. By combining retrieved knowledge with user-specific energy data, EnergyBot provides relevant suggestions to support informed energy management decisions. Together, these features make EnergyBot an intelligent assistance system that improves user awareness, simplifies access to energy-related information, and encourages efficient electricity consumption.

IV. RESULTS

A. Implementation

The system is a web application whose components are listed in Table I. The source code is available at github.com/Nandini-Sarala/AI-Powered-Energy-Optimizer.

TABLE I. IMPLEMENTATION TECHNOLOGIES

Component	Technology and purpose
Front end	React [4] with Vite; dashboard, appliances, chatbot
Back end	Flask [5] REST APIs; JWT authentication
Database	MongoDB Atlas; users, appliances, logs, wallet
Retrieval	Atlas Vector Search [6]; Hugging Face embeddings [7]
Language model	Groq-hosted model [8]
Prediction	Python, scikit-learn; Random Forest
Weather	OpenWeatherMap; temperature, humidity
IoT simulation	ESP32 [9] in Wokwi [10]; Localtunnel

B. Dashboard

The dashboard in Fig. 3 shows units consumed today, current cost, the predicted monthly bill, the final bill after discount, billable units, a daily, weekly, and monthly consumption trend, current weather, and the active KERC tariff of ₹5.80/kWh together with the entitlement. The dashboard also provides navigation options for appliance management, energy-saving recommendations, and user settings. It displays reward points earned by users to encourage energy-efficient consumption habits. The weather information helps users understand how environmental conditions may influence electricity usage, while tariff details support better cost management. Additionally, the interactive interface allows users to refresh data and access important energy-related information in a centralized view.



Fig. 3. Dashboard with key metrics, trend, weather, and tariff.

C. IoT Simulation

The IoT layer is simulated as shown in Fig. 4. In Wokwi, an ESP32 board with a push button and an LED with a current-limiting resistor represents an appliance and its manual override. The controller polls the backend every 3 s through a Localtunnel URL. If Wi-Fi is lost, the simulator script caches local button presses and synchronizes them to the cloud database after reconnection.

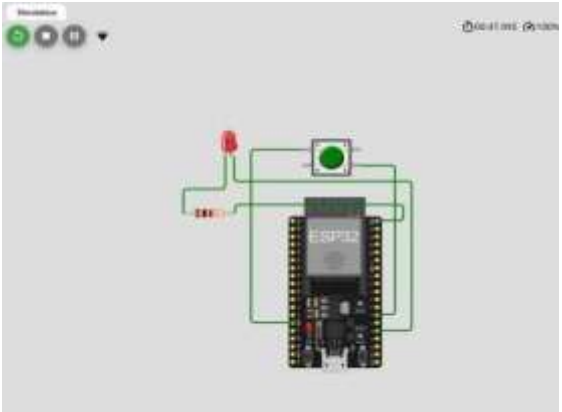


Fig. 4. ESP32 appliance-control simulation in Wokwi.

D. Recommendation Model Performance

Table II reports overall performance on the 2,206 held-out records, and Table III gives per-class precision (P) and recall (R). Both models reach about 88–89% test accuracy and lose under two points

between training and test. The decision tree (88.98%) is marginally ahead of the Random Forest (88.40%), but the two differ markedly on the minority class.

TABLE II. OVERALL CLASSIFICATION PERFORMANCE

Model	Train acc.	Test acc.	Macro F1	Weighted F1
Decision tree	90.39%	88.98%	0.73	0.88
Random Forest	89.94%	88.40%	0.64	0.86

TABLE III. PER-CLASS RESULTS ON THE TEST SET

Class	Support	Decision tree (P / R)	Random Forest (P / R)
Delay_Load	312	0.79 / 0.68	0.80 / 0.69
No_Action	1,746	0.92 / 0.96	0.90 / 0.98
Suggest_OFF	148	0.63 / 0.45	0.87 / 0.14

Suggest_OFF makes up only 6.73% of the data, and the Random Forest detects just 14% of its cases (precision 0.87), whereas the decision tree recovers 45%. The most influential features in the Random Forest are the load ratio (0.214), usage hours (0.118), the Essential and Non-essential priority indicators (0.096 and 0.082), the simultaneous load (0.079), and the temperature (0.047).

E. Reward and Bill Example

Table IV works through the reward rule. The previous July consumption is 776.589 kWh and the projected July consumption is 759.74 kWh. The difference is 16.849 kWh (16.85 kWh to two decimals), which contains one complete 10 kWh block, so one point is earned. Applied to a predicted bill of ₹5,015.63, it gives a final bill of ₹5,014.63.

TABLE IV. WORKED REWARD EXAMPLE

Parameter	Value
Previous July consumption	776.589 kWh
Projected July consumption	759.740 kWh
Saving	16.849 kWh
Complete 10 kWh blocks	1
Reward points earned	1 point (₹1)
Predicted bill	₹5,015.63
Final bill after reward	₹5,014.63

V. DISCUSSION

A. Interpretation of Findings

The small gap between training and test accuracy indicates limited overfitting. On the minority class, the deployed Random Forest rarely issues a wrong Suggest_OFF advice but misses many valid ones, whereas the decision tree trades some precision for higher recall. The feature importances show that the model relies on load, usage, priority, and weather as intended. Because the data are synthetic and the labels follow threshold rules, these figures show how well the model recovers the project's rules, not the savings a household would achieve.

B. Limitations

The training data are synthetic, derived from three homes, and the labels follow rule-based thresholds, so the reported accuracy should be validated on measured household data. The class distribution is imbalanced, and the deployed Random Forest has low recall on Suggest_OFF. Appliance usage is entered or switched by the user in a simulation rather than measured by smart meters, and the ESP32 is simulated in Wokwi, so the prototype does not establish safe switching of real mains-powered appliances. The reward points are not a utility discount, the 200-unit alert is not an eligibility decision, and tariff and scheme rules should be verified against official sources. EnergyBot's answers depend on the coverage of its knowledge base and on the availability of the external database and model services. Response times and chatbot answer quality have not yet been measured systematically.

VI. CONCLUSION

This paper presented an AI-powered energy consumption optimizer that predicts bills under the Karnataka Gruha Jyothi tariff rules, allocates a daily energy budget by appliance priority, rewards savings against the previous year, recommends appliance actions with a Random Forest, and answers questions through a RAG chatbot. A simulated ESP32 controller demonstrates the link between software decisions and appliance outputs. The Random Forest reached 88.40% test accuracy on synthetic data, and further validation with measured data and a pilot deployment are the next steps. The proposed system provides an intelligent and efficient approach to monitoring and optimizing household energy consumption using AI.

Future work includes integrating smart meters and physical ESP32 hardware with properly rated switching components, validating the models on real household data, handling class imbalance (for example, through class weighting), comparing deep forecasters such as the LSTM reported in [3], updating tariffs from verified sources, adding multilingual (including Kannada) chatbot support and a mobile application, and running a pilot with real households.

REFERENCES

[1] N. Raj, S. Bhardwaj, A. Sinha, and A. L. Yadav, "AI-powered energy consumption optimization for smart homes using IoT," in *Proc. Int. Conf. Comput. Intell. Commun. Appl. (ICCICA)*, Samalkha, India, 2024, pp. 231–236, doi: 10.1109/ICCICA60014.2024.10585239.

[2] P. Biswas, A. Rashid, A. Biswas, M. A. A. Al Nasim, S. Chakraborty, K. D. Gupta, and R. George, "AI-driven approaches for optimizing power consumption: A comprehensive survey," *Discover Artif. Intell.*, vol. 4, no. 1, Art. no. 116, 2024, doi: 10.1007/s44163-024-00211-7.

[3] A. M. Durrani, A. U. Asar, A. Aziz, W. Khan, M. Z. Yousaf, U. Farooq, M. Altayeb, and M. S. Geremew, "AI-driven optimization of energy consumption and demand response in smart homes," *Energy Explor. Exploit.*, vol. 44, no. 3, pp. 1382–1419, 2026, doi: 10.1177/01445987251403607.

[4] Meta Open Source, "React documentation." [Online]. Available: https://react.dev

[5] Pallets, "Flask documentation." [Online]. Available: https://flask.palletsprojects.com

[6] MongoDB, Inc., "MongoDB Atlas Vector Search documentation." [Online]. Available: https://www.mongodb.com/docs/atlas/atlas-vector-search/

[7] Hugging Face, "Hugging Face documentation." [Online]. Available: https://huggingface.co/docs

[8] Groq, Inc., "GroqCloud documentation." [Online]. Available: https://console.groq.com/docs

[9] Espressif Systems, "ESP32 series of SoCs." [Online]. Available: https://www.espressif.com/en/products/socs/esp32

[10] Wokwi, "Wokwi documentation." [Online]. Available: https://docs.wokwi.com