

Energy Personalized Recommender for the PowerEye Home Energy Management System

Reem Hejazi*, Sofianiza Abd Malik, Anees Ara, Layla Alfawzan
College of Computer and Information Science, Prince Sultan University, Riyadh, Saudi Arabia

Abstract—The Energy Personalized Recommender (EPR) is a novel backend module designed to optimize energy management within home energy management systems (HEMSs). Existing research explores various energy optimization strategies but is often constrained by synthetic datasets, simulation-based validation, or limited personalization. To address these limitations, EPR is designed for real-world deployment and personalization. By employing machine learning algorithms, the EPR analyzes appliance-level energy consumption in real time and integrates various algorithms to enhance user interaction with energy usage data. These algorithms include the Energy Goal Algorithm, which allows users to set and monitor monthly energy cost goals, and the Peak Time Algorithm, which advises on shifting usage to off-peak times. The Phantom Mode Algorithm identifies low-power or standby appliances, promoting energy conservation, while the Energy Baseline Algorithm predicts hourly energy usage to detect deviations. In evaluating the system, k-means clustering for phantom appliances achieved silhouette scores of 0.478 to 0.954, indicating reliable performance. Baseline prediction models showed RMSE values between 0.003 and 1.526, depending on the appliance. A single-day evaluation involving three participants showed mixed consumption changes: two participants recorded decreases, while the third recorded an increase, resulting in a 19.74% aggregate increase. These preliminary findings provide a foundation for advancing personalized energy management and motivate longer controlled trials to determine efficient and sustained energy savings through EPR systems.

Keywords—Home energy management systems; machine learning; energy goal; peak time; phantom mode; energy baseline; k-means clustering; XGBoost forecasting; energy sustainability

I. INTRODUCTION

Recommender systems have become essential tools in various domains, offering personalized suggestions that enhance user experience and decision-making. In energy management, these systems are crucial in promoting energy efficiency and sustainability. Home energy management systems (HEMSs) are designed to collect and analyze energy data from household appliances, providing users with valuable insights into their energy usage patterns [1].

Current research in the field of HEMS has explored not only basic energy monitoring but also the integration of recommender systems to provide personalized insights [2], [3]. Several studies have proposed hybrid energy recommender systems that combine multiple optimization strategies to support household energy management [4], [5], including demand-side management in smart grids [6] and energy consumption forecasting using machine learning approaches such as RNN-LSTM [7]. While these approaches demonstrate the benefits

of hybridization, they typically focus on a limited subset of objectives. In contrast, our proposed HEMS, PowerEye, introduces an Energy Personalized Recommender (EPR) designed as a modular hybrid framework that simultaneously integrates four complementary optimization algorithms: cost-based goal tracking, peak-time shifting, phantom energy detection, and real-time baseline forecasting. This unified design enables more comprehensive and actionable recommendations, thereby enhancing both energy efficiency and sustainability.

This paper aims to explore the development and impact of EPR within the PowerEye system. The contributions of the proposed work are as follows:

- We conduct a detailed state-of-the-art literature review of energy recommender system technologies.
- We propose a hybrid-approach EPR framework as part of the HEMS, PowerEye. This system demonstrates personalized recommendations that can optimize energy usage in households.
- We deploy PowerEye using 23 commercial smart plugs across three households, collecting 111 days of appliance-level data for model development and evaluation. A separate single-day experiment examines consumption changes following recommendations.
- We develop the EPR algorithms using k-means clustering and XGBoost. These algorithms are used to recommend and predict optimal energy usage.
- We conduct an evaluation of the experiments, including phantom clustering accuracy, baseline forecasting accuracy, energy consumption analysis, and user experience. These results assess the effectiveness of personalized recommendations.

The paper is organized as follows: Section II provides a comprehensive review of the literature on energy recommendation systems; Section III outlines the PowerEye system architecture and related details; Section IV describes the simulation setup and deployment of the proposed system; Sections V and VI present the results and discussion, respectively; and finally, Section VII concludes the paper with future recommendations.

II. BACKGROUND

The following section provides a brief overview of the background of energy recommender systems and a comparative analysis of related work.

*Corresponding author.

A. Types of Energy Recommender Systems

The primary aim of HEMSs is to enable users to monitor and optimize their energy consumption through personalized recommendations. Various models are employed for energy optimization in recommender systems, including the following:

1) *Content-based*: Suggests items that are similar to those a user has liked in the past by analyzing item features [8]. It builds a user profile based on features of items previously rated by the user, such as keywords from item descriptions.

2) *Collaborative filtering*: Makes suggestions based on the behaviors of users with similar preferences [9]. This approach analyzes the rating history of users and calculates the similarity between them. Recommendations are then made by suggesting items that other users with similar tastes have liked in the past.

3) *Knowledge-based*: Suggests items based on predefined rules and domain knowledge, ensuring they meet specific user requirements and preferences [10]. Unlike systems based on rankings or item features, this approach evaluates items or actions against a set of criteria defined by the user.

4) *Case-based*: Uses historical examples or past cases to inform recommendations [11]. It retrieves relevant cases from memory to solve new problems by adapting previous workflows to fit current conditions. Once a suitable case is identified, it is modified for the new situation and reused.

5) *Context-based*: Suggests items by considering contextual factors such as time, location, or other external conditions [12]. By adapting to the user's changing needs and situations, context-based systems aim to provide more relevant and timely recommendations that align with the user's current circumstances.

6) *Hybrid*: Combines two or more recommendation techniques to improve the accuracy and relevance of recommendations.

B. Literature Review

Jimenez-Bravo et al. [10] proposed a multi-agent system for smart homes that optimizes energy usage by coordinating agents. These agents, built with a Python library, collect real-time consumption data, track electricity prices, and offer personalized energy-saving recommendations. The system uses knowledge-based and utility-based recommendations to provide financial incentives for shifting energy use. Though promising for addressing energy optimization issues in the European Union, the system was only tested in a theoretical case study, not real-world settings.

Eirinaki et al. [4] proposed a framework to help households improve energy efficiency by providing real-time recommendations for appliance use. The framework uses non-intrusive load monitoring (NILM) to disaggregate individual appliance consumption from main meter data and applies association rule mining and sequential pattern mining algorithms to extract usage patterns. These patterns are used to build personalized user profiles and generate recommendations for energy savings. The study found that the framework could reduce energy consumption by 2–17%.

Gonzalez-Briones et al. [11] present a multi-agent system architecture designed to optimize temperature and energy

consumption in an office building. By analyzing sensor data and external information, the system establishes temperature patterns that align with both energy efficiency and user comfort. The GAIA methodology [13] guided the design and development of this system, incorporating algorithms for savings, machine learning, case-based reasoning, and artificial neural networks. The findings demonstrate the system's effectiveness, achieving average energy savings of 41% in the experimental group offices compared to the control group.

Zhang et al. [14] introduce an electricity plan recommender system (EPRS) designed to offer personalized recommendations to residential customers. Leveraging neighborhood-based collaborative filtering, the EPRS demonstrates superior performance compared to existing strategies in terms of recommendation accuracy. The J-wEUC similarity metric, a novel approach developed for this study, effectively addresses the challenge of missing features. By providing customers with predicted ratings of various tariff plans, the EPRS facilitates the widespread adoption of flexible pricing tariffs.

Sardianos et al. [5] present REHAB-C, a context-aware recommendation system designed to assist users in transforming their energy habits. REHAB-C utilizes a rule-based system to detect user habits and recommend actions based on the user's context and goals. Additionally, the system incorporates collaborative filtering to leverage feedback from multiple users, enhancing the recommendations. REHAB-C was evaluated in real-world office scenarios and demonstrated a significant reduction in energy consumption, up to 50%.

Zhang et al. [15] present a customized electricity retailing plan recommendation strategy that assists customers in selecting economical retailing plans. The strategy utilizes customer electricity usage patterns to recommend cost-effective plans without requiring full daily load profile records. The methodology involved two-stage DLP clustering, correlative feature selection, decision tree classification, and typical plan rank and recommendation plan rank calculation. The study identified correlative features between load profile features and DLP classes, providing insights for both plan selection and further investigations into the retail market, such as tariff design.

Rodriguez et al. [16] present a novel methodology that utilizes various algorithms to analyze energy consumption patterns in household refrigerating appliances, leading to tailored energy-saving recommendations. Their approach incorporates cycle counting to determine operating cycles and a 7-day centered moving average for smoothing energy data and estimating base consumption. By employing k-means clustering with Dynamic Time Warping, the study categorizes daily energy profiles. The findings demonstrate that the probability density estimation method effectively identifies refrigerators with anomalously high energy consumption, enabling personalized recommendations.

A comparison between these existing energy recommendation systems is shown in Table I.

The literature presents diverse frameworks for energy management, including multi-agent systems, real-time recommendations, and personalized plan suggestions. These studies leverage approaches like collaborative filtering, machine learning, and clustering to optimize energy use and provide

TABLE I. COMPARISON OF EXISTING ENERGY RECOMMENDATION SYSTEMS

	Features	Methods	Limitations
[10]	- Minimize peak consumption - Utilize smart plugs - Use online electricity prices	- Multi-agent - Knowledge-based - Utility-based	- Not real-world settings - Limited to the Spanish electricity market - Supports only one smart plug
[4]	- Create appliance profiles - Switch on/off suggestions - Interactive energy suggestion interface	- HMM model - Combinatorial optimization - k-means clustering	- Ignore inter-appliance associations - Omit weather data - Omit environ. psychology
[11]	- Automatic temperature adjustment - Data-driven HVAC temperature control - Case-based reasoning	- Multi-agent - GAIA - Case-based	- Lacks comfort parameters with energy savings - Insufficient data reporting - Lacks granular insights
[14]	- Recommend cost-effective electricity plans - Offer predicted ratings - Appliance on thresholds	- CF filtering - Neighborhood-based - Jaccard Euclidean	- Slow calculation speed - Limited dataset - Limited exploration of the impact on market policies
[5]	- Goal-based system - Incorporate smart sensors - Incorporate users' preferences and feedback	- CF filtering - Rule-based - Context-aware	- No support for setting energy goals - Limited to a single user - No standardized benchmarks
[15]	- Rank cost-effective electricity - Classify daily load profiles - Link DLP classes to a typical plan rank	- k-means clustering - Decision trees - Dominant plan rank selection	Not mentioned
[16]	- Analyze fridge energy use - Evaluate base and relative energy consumption - Identify anomalous energy	- One-way ANOVA - k-means clustering - Probability density estimation	- Limited dataset - Recommender impact has not been evaluated - Does not consider seasonal effects

savings insights. Notable findings show energy reduction potential from 2% to 50%, though many systems are limited by theoretical models, dataset constraints, or lack of real-world validation. Hence, in the following section, we address some of these gaps and propose an energy-personalized recommender for a home energy management system.

III. METHODS

In this section, we discuss the design and architecture of our proposed HEMS, PowerEye, along with its system components, personalized recommendation strategy, and the recommendation process. Finally, we provide an in-depth explanation of the four algorithms used in our EPR: energy goal, peak time, phantom mode, and energy baseline algorithms.

A. PowerEye: The Proposed Home Energy Management System

Our proposed HEMS, PowerEye, aims to build a user-friendly mobile application that empowers users to effortlessly monitor their energy usage at the appliance level. Users can set energy consumption goals, easily manage devices and rooms, and receive real-time data insights to make informed decisions. Even individuals with minimal technical knowledge will find it easy to set up the smart plugs and devices, without needing technical support. With interactive charts and real-time data at their fingertips, PowerEye offers an engaging, intuitive experience, making energy efficiency achievable for everyone.

Existing energy recommendation systems often focus on one specific aspect of energy optimization, such as reducing peak consumption, providing switch-on/off recommendations, or minimizing electricity costs. These systems tend to rely on single methods like collaborative filtering or rule-based approaches, which limit their effectiveness by not offering comprehensive solutions.

In contrast, PowerEye's EPR further enhances this by using a hybrid approach to energy optimization. It helps users set cost-saving goals, recommends peak-time savings, detects phantom power modes, and identifies excessive energy usage, addressing multiple aspects of energy efficiency in one solution.

B. System Architecture

The proposed home energy management system, PowerEye, has a modular architecture with system functionalities distributed among different modules or components, as shown in Fig. 1. Each module interacts with the others using specified protocols and APIs.

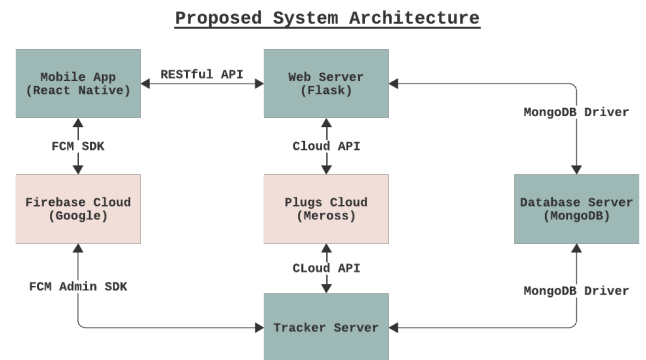


Fig. 1. System architecture design diagram.

The proposed PowerEye system is broken down into the following components:

1) *Database server*: This server stores and manages data such as user profiles, appliance information, and energy consumption records. MongoDB is utilized for its flexibility and scalability [17], making it ideal for efficiently handling time-series and semi-structured IoT data.

2) *Plugs cloud (meross)* [18]: The Plugs Cloud component acts as a bridge between the application and Meross smart plugs. It enables communication through an unofficial API provided by Alberto Geniola [19], enabling real-time information retrieval and remote control.

3) *Web server*: This server uses the Flask framework to handle HTTP requests via RESTful APIs, manage user authentication, and serve as the primary point of interaction between users and the application [20]. Flask [20] was selected for its lightweight nature and ease of integration with data-driven services.

4) *Firebase cloud* [21]: Firebase Cloud Messaging (FCM) is responsible for sending push notifications to mobile devices. It acts as a communication bridge between the Tracker Server and the mobile application. It ensures that users are promptly informed of critical events, such as reaching energy goals or receiving recommendations.

5) *Tracker server*: This is the backbone of the system, managing power data from Meross Cloud [18], storing device details in MongoDB [17], tracking events like energy goals, and delivering personalized recommendations and push notifications through FCM [21]. It centralizes backend operations, processes device data, and offers valuable insights.

6) *Mobile application*: The React Native-based mobile app offers a platform-independent interface for smart plug control [22]. Users can view device status, consumption data, and recommendations, and control devices remotely. React Native streamlines cross-platform app development with a single codebase, saving time and ensuring a consistent experience on Android and iOS.

C. Energy Personalized Recommender Design

In our proposed recommender system, we focus on four aspects of energy optimization: energy goal, peak time, phantom mode, and energy baseline.

1) *Energy goal*: Users can set monthly energy consumption goals based on cost. The system then calculates the energy equivalent to the specified cost. Users are then notified when their energy consumption hits specific percentage thresholds, ranging from 25% to 200%.

2) *Peak time*: Avoiding appliance usage during peak hours lowers emissions, conserves resources, cuts infrastructure costs, and supports renewable energy [23]. We will advise users to postpone appliance use during peak times.

3) *Phantom mode*: This refers to appliances consuming electricity even when they are turned off but still plugged in [24]. This happens because they stay in standby mode to respond quickly when needed. In this scenario, we will advise users to turn off appliances in phantom mode.

4) *Energy baseline*: Our system provides users with insights into their energy consumption patterns by sending alerts when they exceed their daily baseline. Each appliance in the system is equipped with its own model, allowing it to forecast energy consumption for the upcoming hour.

Based on the above aspects, we can conclude that our EPR employs a hybrid approach. It can be considered context-based as it uses contextual factors such as time, appliance status, power, and energy readings. Additionally, it qualifies as knowledge-based because it relies on user preferences and predefined rules for energy goals and peak times. Finally, it has content-based elements as it utilizes historical user energy patterns to predict future usage and issue alerts for excessive consumption.

D. Recommendation Process

The Tracker Server collects power readings from each Meross smart plug at one-minute intervals [18]. The EPR is designed using a modular architecture, in which the four optimization modules (energy goal tracking, peak-time shifting, phantom energy detection, and baseline forecasting) operate as independent components. Each module accesses the required input data from a shared data layer and does not rely on the outputs of other modules.

As the Tracker Server gathers data, it invokes the relevant EPR modules independently based on predefined conditions and scheduling logic, enabling the system to generate personalized recommendations for the user. The Tracker Server then sends these recommendations to the user's mobile application via Google Firebase [21]. This process is illustrated in Fig. 2.

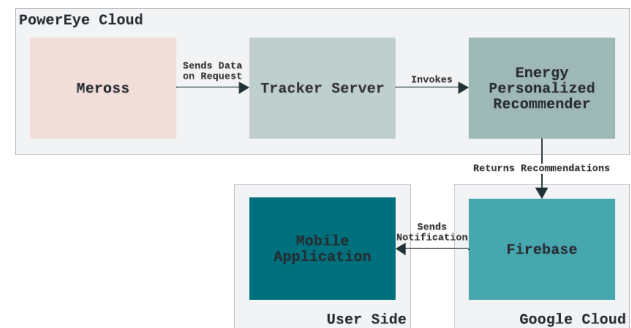


Fig. 2. Recommendation process.

The data used include user information, appliance details, power readings, and energy consumption. To calculate daily energy consumption, Eq. 1 is applied:

$$E(kWh) = \sum_{i=1}^n P_i \Delta t = \sum_{i=1}^{24 \times 60} \frac{P_i}{1000} \times \frac{1}{60} \quad (1)$$

Energy consumption is calculated by converting power from watts to kilowatts and multiplying it by the time interval, which is set at one minute (1/60 hour). The daily energy consumption values are then aggregated and stored with the unit kWh.

E. Energy Goal Algorithm

Users can set a monthly cost goal for their energy consumption. In Saudi Arabia, the energy cost is calculated by multiplying the energy consumption by the Residential Consumption Tariff (RCT), as specified on the official website [25]. The energy goal can then be calculated by rearranging Eq. 2 as shown in Eq. 3.

$$Cost(SAR) = E \times RCT = E \times \frac{0.8}{100} \quad (2)$$

$$E(kWh) = Cost \times \frac{100}{0.8} \quad (3)$$

The `check_goal()` function, provided by the EPR module, computes the percentage of the user's energy goal that has been reached. This percentage is determined by dividing the current month's energy consumption by the user's set goal, rounding down to the nearest quarter, and ignoring any quarter greater than 2. The result is then converted into a percentage out of 100. Upon reaching each specified threshold, the server sends a notification. Flags are used to prevent multiple notifications for the same percentage within the current month. The steps for this process are provided in Algorithm 1:

Algorithm 1 Check Goal

```
1: Input: month_energy (float), goal (float)
2: Output: int

3: function CHECK_GOAL(month_energy, goal)
4:   percentage  $\leftarrow$  month_energy / goal
5:   rounded  $\leftarrow \lfloor \text{percentage} * 4 \rfloor / 4$ 
6:   if rounded  $\leq 2$  then
7:     return int(rounded * 100)
8:   else
9:     return 0
10:  end if
11: end function
```

F. Peak Time Algorithm

The `check_peak()` function, provided by the EPR module, uses straightforward logic. It checks whether the appliance is turned on during peak times and whether its energy type is categorized as shiftable. For more details on appliance types, refer to Appendix A. In Saudi Arabia, peak time is recognized by the government as 1–5 p.m., as stated on the government website [26]. If the function returns true, the server sends a notification. Flags are used to prevent additional notifications for the same appliance during the current day. The steps for this process are provided in Algorithm 2.

G. Phantom Mode Algorithm

Phantom appliances can exist in three distinct states: on, off, and phantom. The primary objective of this algorithm is to identify the phantom state. This is achieved by clustering appliance power data into two groups: `on_cluster` and `phantom_cluster`. The off state is intentionally included within the `phantom_cluster`, as phantom consumption

Algorithm 2 Check Peak

```
1: Input: cur_hour (int), status (boolean), e_type (EType),  
   types (list)
2: Output: boolean

3: PEAK_START  $\leftarrow$  13
4: PEAK_END  $\leftarrow$  17
5: function CHECK_PEAK(cur_hour, status, e_type, types)
6:   is_peak  $\leftarrow$  PEAK_START  $\leq$  cur_hour  $<$  PEAK_END
7:   return is_peak and status and e_type in types
8: end function
```

typically occurs at very low power levels close to zero. Distinguishing between off and phantom states is then performed using the appliance status (ON/OFF), enabling accurate separation without requiring additional clustering complexity.

The dataset is extracted from the PowerEye database, containing timestamped appliance-level power readings. Only power values are used for clustering. The preprocessing steps are as follows:

- **Null Values:** These can be dropped as there is no need for imputation in this context.
- **Dataset Size:** A sample size of 1,000 readings was selected as a practical balance between data collection requirements and computational cost. The resulting clustering performance is reported through appliance-level silhouette scores (Table IV); sensitivity to sample size remains to be evaluated.
- **Duplicate Values:** Rather than being removed, duplicates are assigned weights to preserve their frequency information while avoiding bias in clustering.

For clustering, k-means with $k = 2$ is employed [27]. This choice is motivated by the univariate nature of the data (power readings only) and the expectation of two dominant consumption patterns: active and near-zero power. Compared to more complex methods such as DBSCAN or Gaussian mixture models, k-means provides a lightweight and deterministic solution suitable for real-time deployment [28]. Density-based methods like DBSCAN require sensitive parameter tuning and are less stable in one-dimensional data, while probabilistic models such as GMM introduce unnecessary complexity without significant performance gains for this task. Similar approaches have been effectively used in prior work [4]. A publicly available Jupyter notebook provides more detailed information [29]. One model is created for each appliance and stored on PowerEye's server, where it is used within the `check_phantom()` function.

The `check_phantom()` function, provided by the EPR module, first checks whether a cluster model already exists for the appliance. If the appliance is turned on, it uses the model to predict the cluster of the current power reading. The algorithm compares the predicted cluster with the cluster associated with a power reading of 0. If they match, it indicates that the appliance is in phantom mode, and the function returns True. When the function returns True, the server sends a notification. In cases where the clusters are not well separated, flags are used to avoid sending excessive notifications in a short period. The steps for this process are detailed in Algorithm 3.

Algorithm 3 Check Phantom

```
1: Input: model (clusterModel), power (float), status  
   (boolean)  
2: Output: boolean  
  
3: function CHECK_PHANTOM(model, power, status)  
4:   if model and status then  
5:     predicted_labels  $\leftarrow$  model.predict([0, power])  
6:     return predicted_labels[0] == predicted_labels[1]  
7:   end if  
8:   return False  
9: end function
```

H. Energy Baseline Algorithm

The final EPR step alerts users of potential excessive usage by forecasting appliance-level energy consumption. If actual consumption deviates from predictions, users are notified that their usage exceeds expected levels. This approach helps users monitor and reduce energy usage. Machine learning models predict hourly consumption based on historical data, setting an expected energy limit. Users are notified if they exceed this limit.

To prevent data leakage and preserve temporal causality, all preprocessing and model evaluation steps strictly respect the chronological order of observations. The preprocessing pipeline is defined as follows:

1) *Energy conversion and hourly resampling:* Power readings are converted into energy using Eq. 1 with $n = 60$, then resampled hourly, generating more data for better prediction accuracy and quicker energy-saving decisions.

2) *Null values:* Rows with more than 30 null values (before resampling) are dropped.

3) *Normalization:* A standard scaler is used to handle large magnitude differences. The dataset is split 80/20 into training and test sets before normalization to keep the model blind to the test data.

4) *Feature engineering:* To transform the time series into a regression problem, we decompose the timestamp into features like hour, month, year, day, and whether it is a weekend or nighttime. We then add rolling statistics (e.g., average, standard deviation) to capture trends and variations. Lastly, we include seven lag features to balance predictive power and computational efficiency.

5) *Dataset splitting:* Half of the testing dataset is reallocated for validation, resulting in a new ratio of 80% for training, 10% for validation, and 10% for testing (0.8:0.1:0.1). This ensures that all training observations precede validation observations, and validation observations precede test observations, preventing future information from leaking into model training.

6) *Feature selection:* The most relevant features are identified by assessing their impact on the model's predictions. Using XGBoost with default hyperparameters [30], the top 15 features with the highest importance scores are selected to improve model performance and reduce overfitting.

For forecasting, the XGBoost model—short for eXtreme Gradient Boosting—is employed using time-series cross-validation with walk-forward validation to maintain the dataset's temporal structure [30]. This approach trains on historical data and validates subsequent points, avoiding look-ahead bias and providing realistic performance assessments. Hyperparameter tuning is essential since XGBoost has numerous hyperparameters. The tuning process followed Jain's approach [31], using Mango Tuner instead of grid search cross-validation. Mango Tuner is a state-of-the-art gradient-free optimizer that offers faster performance and greater flexibility in setting objective functions [32]. The objective function employs time-series cross-validation (walk-forward) and aims to minimize mean squared error (MSE).

After finding the optimal hyperparameters, the model is trained using these parameters and then utilized to predict the energy consumption for the following hour, which serves as a baseline. An actual value is considered abnormal if it significantly deviates from this baseline. A normalized relative MAE score is used, where values exceeding a threshold of 0.75 are flagged as excessive usage [33]. This threshold was chosen as a conservative operating point to balance sensitivity and false-positive rates in practical deployment. Lower thresholds (e.g., 0.6–0.7) were observed to increase false alerts due to short-term fluctuations and sensor noise, while higher thresholds (e.g., above 0.8) reduced sensitivity to meaningful deviations. The score is calculated by taking the absolute difference between the predicted and actual values, dividing it by the mean absolute error (MAE) [33], and then normalizing the result to compare against the fixed threshold. The formulas are presented in Eq. 4, 5, and 6.

$$MAE = \frac{\sum_{i=1}^n |Predicted_i - Actual_i|}{n} \quad (4)$$

$$Score_i = \frac{|Predicted_i - Actual_i|}{MAE_{total}} \quad (5)$$

$$NormalizedScore_i = \frac{Score_i - \min(Score)}{\max(Score) - \min(Score)} \quad (6)$$

The optimal hyperparameters, along with the MAE, maximum score, and minimum score obtained during the tuning phase, are stored on PowerEye's server for later use in the `check_baseline()` function. This function, part of the EPR module, applies the same preprocessing steps described earlier to all power readings up to the current hour. It then splits the data to train the XGBoost model and tests it on the most recent reading. The function calculates the normalized relative MAE and returns True if the actual energy consumption exceeds the predicted value and the score surpasses the 0.75 threshold. If True, the server sends a notification. Flags are used to prevent sending multiple notifications for the same appliance within the same day. The process is outlined in Algorithm 4, with detailed steps available in our Jupyter notebook [29].

IV. POWEREYE SIMULATION

The following section includes details about the deployment and experimental setup.

Algorithm 4 Check Baseline

```
1: Input: powers (time-series), mae (float), max_score  
   (float), min_score (float), params (dict)  
2: Output: boolean  
  
3: function CHECK_BASELINE(powers, mae, max_score,  
   min_score, params)  
4:   count null values for each hour  
5:   drop any row with more than 30 null values  
6:    $df["energy"] \leftarrow (powers/1000) * (1/60)$   
7:    $df \leftarrow df.resample("H").sum()$   
8:    $df \leftarrow normalize(df)$   
9:    $df \leftarrow feature\_engineering(df)$   
  
10:   $y \leftarrow df["energy"]$   
11:   $X \leftarrow df.drop("energy")$   
12:   $y\_train, y\_test \leftarrow y[:-1], y[-1]$   
13:   $X\_train, X\_test \leftarrow X[:-1], X[-1]$   
  
14:   $model \leftarrow XGBRegressor(params)$   
15:   $model.fit(X\_train, y\_train)$   
16:   $pred \leftarrow model.predict(X\_test)$   
  
17:   $r\_mae \leftarrow abs(pred - y\_test) / mae$   
18:   $nr\_mae \leftarrow (r\_mae - min\_score) / (max\_score -$   
    $min\_score)$   
19:  return  $y\_test > pred$  and  $nr\_mae > 0.75$   
20: end function
```

A. Proposed System Deployment

The deployment of PowerEye involved configuring both hardware and software systems. All deployment configuration details for the proposed system are summarized in Table II.

TABLE II. POWEREYE DEPLOYMENT CONFIGURATIONS

Component	Configuration
Database	MongoDB Atlas on AWS (Bahrain Region), M0 Sandbox cluster, 512 MB storage, shared RAM
Firebase (FCM)	Configured for iOS and Android with secure authentication keys, APN for iOS, SHA certificate for Android
Web Server	AWS EC2 t3.micro instance, Ubuntu Server 22.04 LTS, 30 GB SSD storage, 2 vCPUs, 1 GiB RAM
Tracker Server	Orange Pi 5 SBC, Rockchip RK3588S 8-core processor, 8 GB LPDDR4 RAM, 32 GB microSD storage, Ubuntu 22.04.2 LTS
Mobile Application	React Native with Expo, iOS and Android deployment, Apple Distribution certificates, ad hoc provisioning profiles for push notifications and internal distribution

B. Experimental Setup

A total of 23 commercial smart plugs were purchased and distributed among three participants with different characteristics, as detailed in Table III.

Data collection for the HEMS lasted for 111 days, initially using temporary cloud-hosted code. Later, the Tracker Server took over the data collection process. The participants installed the PowerEye application on their mobile devices, enabling them to access their accounts and monitor their data. The mobile application experiment was limited to a single day due to time and resource constraints.

TABLE III. EXPERIMENTAL PARTICIPANT INFORMATION

User	Rooms	Family Members	Smart Plugs	Appliance Types
User 1	10	9	8	4 Phantom, 4 None
User 2	8	7	6	1 Phantom, 5 None
User 3	8	5	9	4 Phantom, 5 None

V. RESULTS

The following section discusses the evaluation of the experiments using phantom clustering and baseline forecasting accuracy, user experience, and comparative analysis.

A. Phantom Clustering Accuracy

Scatterplots for the power clusters of all phantom appliances across users are presented in Fig. 3. The quality of the k-means clustering models is evaluated using the silhouette score [34], which quantifies the similarity of data points within their cluster (cohesion) relative to other clusters (separation). Silhouette scores range from -1 to 1, with higher scores indicating better clustering. Table IV provides the silhouette scores, rounded to three decimal places, for all 2-means models of phantom appliances.

TABLE IV. APPLIANCE SILHOUETTE SCORES

User 1			
office_strip_10: 0.572	tv_10: 0.678	tv_11: 0.788	charger_10: 0.637
User 2			
charger_20: 0.503			
User 3			
speaker_30: 0.915	tv_30: 0.609	charger_30: 0.478	office_strip_30: 0.954

B. Baseline Forecasting Accuracy

Line plots illustrating the energy consumption of all appliances across users are presented in Fig. 4. The blue line represents the actual energy consumption, the red dashed line indicates the predicted energy consumption, and the black dots highlight the actual values that significantly deviate from the predicted ones. Since this is a regression problem, Root mean squared error (RMSE) and MAE were used to evaluate the XGBoost models for each appliance. All scores are presented in Table V, rounded to three decimal places. Appliances that did not meet the requirements for the XGBoost model were omitted from the table.

C. Energy Consumption Analysis

We conducted a one-day experiment with three existing users via the PowerEye app (Fig. 5). Table VI compares measured energy consumption on the experiment day with the previous day.

D. User Experience

To evaluate user satisfaction with the PowerEye application, we created a survey that includes several questions

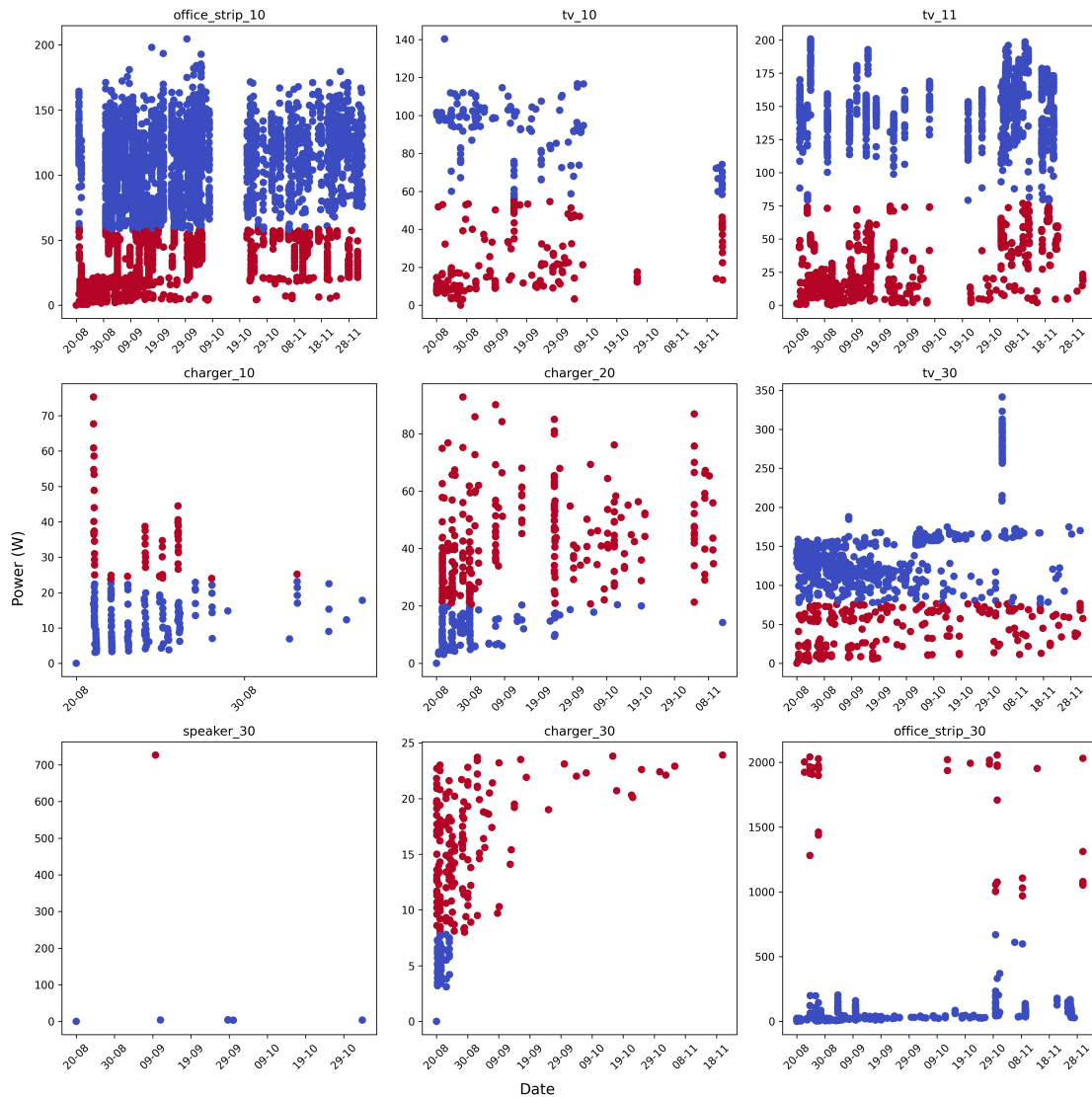


Fig. 3. Phantom appliance power clusters using k-means clustering models.

about their experience. The survey specifically assesses the relevance and helpfulness of the recommendations, as well as how effectively the application aids users in optimizing their energy consumption. Below are the questions related to the EPR and their respective responses, as shown in Fig. 6:

1) *Peak time notifications*: If you received peak time notifications, did the appliance mentioned in the notification match one whose use you could postpone? Did this notification help you optimize your energy consumption?

2) *Goal notifications*: If you received a goal notification, did the percentage mentioned match the percentage displayed in the app? Did this notification assist you in understanding your energy consumption?

3) *Phantom mode notifications*: If you received a phantom mode notification, was your device actually in standby mode? Did this notification assist you in eliminating energy wastage?

4) *Energy baseline notifications*: If you received an energy baseline notification, did you think the appliance mentioned in

the notification was used excessively during that hour? Did it help you better understand your energy consumption?

VI. DISCUSSION

A. Phantom Clustering

As shown in Fig. 3, some of the scatterplots exhibit high separation between clusters, while others do not. We can establish a threshold of 0.5 for our models. In practice, average silhouette values between 0.5 and 0.7 are commonly interpreted as reflecting moderate to good clustering structure, providing a balanced criterion for identifying meaningful clusters while avoiding over-sensitivity to noise. We can also introduce a flag to avoid sending too many notifications in a short period.

There are two cases when the model predicts incorrectly:

- **False Positive**: The model incorrectly classifies an on_state as phantom, leading to potentially annoying notifications. User feedback can help mitigate this.

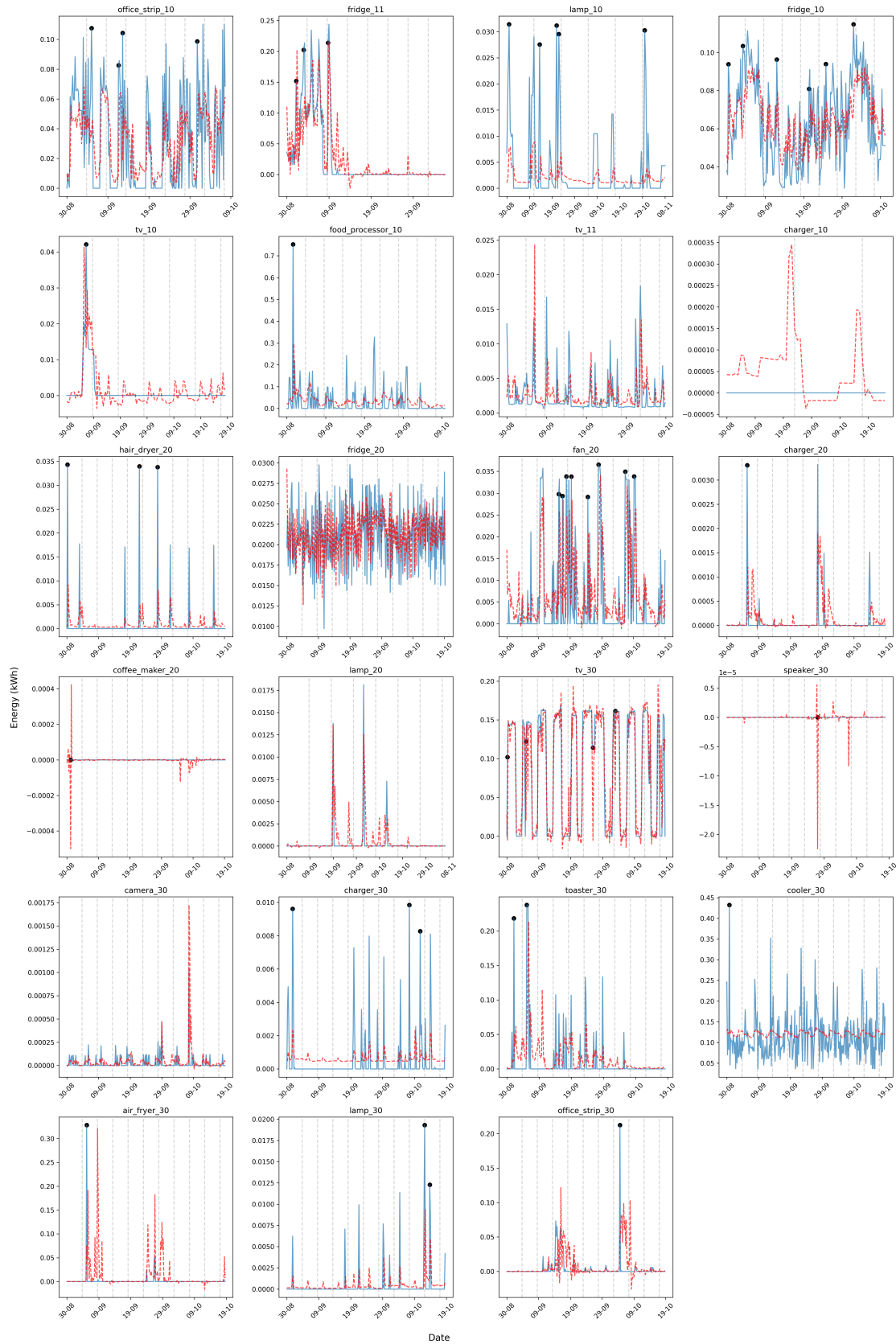


Fig. 4. Actual vs. predicted appliance energy consumption.

TABLE V. APPLIANCE MAE AND RMSE SCORES

User 1					
office_strip_10		fridge_11		lamp_10	
MAE	RMSE	MAE	RMSE	MAE	RMSE
0.788	1.024	0.300	0.603	0.834	1.526
tv_10		food_processor_10		tv_11	
MAE	RMSE	MAE	RMSE	MAE	RMSE
0.060	0.112	0.492	0.904	0.084	0.147
fridge_10		charger_10			
MAE	RMSE	MAE	RMSE		
0.717	0.849	0.017	0.026		
User 2					
hair_dryer_20		fridge_20		fan_20	
MAE	RMSE	MAE	RMSE	MAE	RMSE
0.371	1.086	0.510	0.647	0.437	0.617
coffee_maker_20		lamp_20		charger_20	
MAE	RMSE	MAE	RMSE	MAE	RMSE
0.002	0.011	0.068	0.218	0.055	0.151
User 3					
tv_30		speaker_30		camera_30	
MAE	RMSE	MAE	RMSE	MAE	RMSE
0.249	0.432	0.000	0.003	0.253	0.639
toaster_30		cooler_30		air_fryer_30	
MAE	RMSE	MAE	RMSE	MAE	RMSE
0.647	1.239	0.706	0.953	0.266	0.906
office_strip_30		charger_30		lamp_30	
MAE	RMSE	MAE	RMSE	MAE	RMSE
0.389	0.948	0.478	0.824	0.227	0.589

TABLE VI. ENERGY CONSUMPTION BEFORE AND AFTER THE EXPERIMENT

	Before (kWh)	After (kWh)	Change (%)
User 1	0.802	0.796	-0.75
User 2	0.720	0.682	-5.28
User 3	4.374	5.582	+27.62
Total	5.896	7.060	+19.74

Change = $100 \times (\text{After} - \text{Before}) / \text{Before}$; negative values indicate reductions.

- False Negative: The model incorrectly classifies a phantom as on_state, missing a notification. This reduces energy optimization but has a minimal impact.

B. Baseline Forecasting Accuracy

Forecasting accuracy varied across appliances (Table V). Several models achieved low errors on the standardized consumption scale, including RMSE values of 0.147 for tv_11 and 0.218 for lamp_20, while others showed larger deviations. These results characterize appliance-level prediction accuracy; comparison with a historical-average or persistence predictor is needed to establish the benefit of XGBoost over simpler forecasts.

Forecasting errors can affect excessive-use detection in two ways:

- False Positive: The model wrongly predicts excessive energy use. We can reduce this by limiting notifications to once a day.
- False Negative: The model fails to detect excessive energy use. This missed notification impacts optimization minimally.

Due to the experiment's short duration and resource constraints, we were unable to evaluate additional models. However, more advanced time-series models such as long short-term memory (LSTM) networks or autoregressive integrated moving average (ARIMA) could be considered in future work, as they are designed to capture temporal dependencies and may improve forecasting accuracy. These models, however, require higher computational resources and careful tuning to avoid overfitting, particularly when data are limited.

C. Energy Consumption Analysis

Table VI shows mixed outcomes: two participants reduced consumption, but the third participant's increase raised total consumption from 5.896 to 7.060 kWh (19.74%). Thus, the experiment did not demonstrate an overall energy reduction. The single-day intervention, small sample, and absence of a control group prevent attributing these changes to EPR rather than ordinary day-to-day variation. Longer controlled trials are needed to evaluate sustained energy savings.

D. User Experience

The majority of users reported satisfaction with PowerEye's EPR notifications (Fig. 6). While precision/recall metrics and user engagement rates are commonly used to evaluate recommendation effectiveness, the limited number of participants

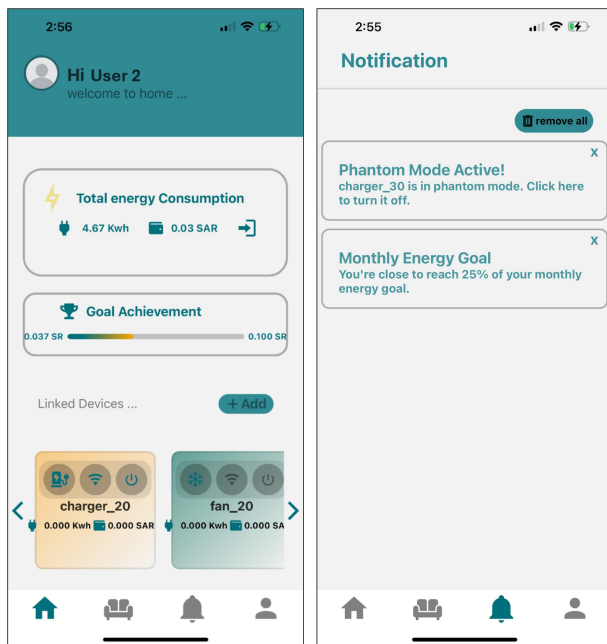


Fig. 5. PowerEye mobile application interfaces.

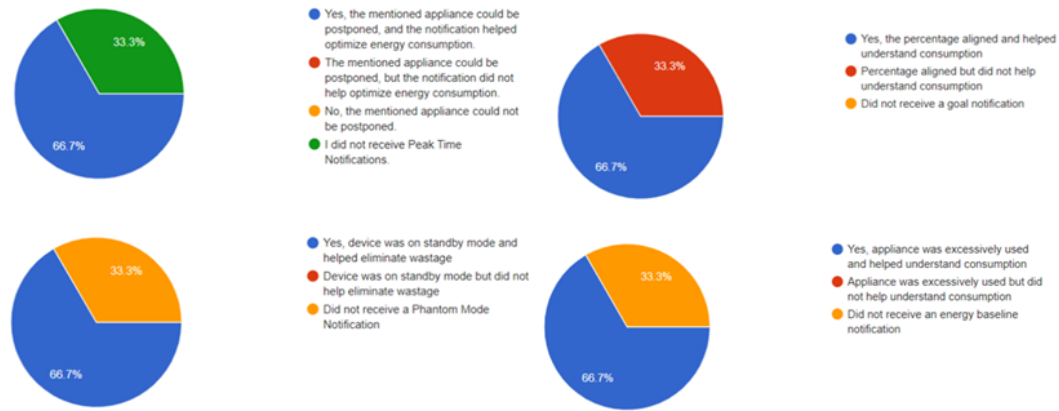


Fig. 6. User experience evaluation responses.

TABLE VII. EXISTING ENERGY RECOMMENDATION SYSTEMS VS. THE PROPOSED EPR

	Type	Multi-agent system	Uses price data	User in control	Peak time savings	Anomaly detection	Phantom mode detection
[10]	Hybrid	✓	✓	✓	✓		
[4]	Hybrid			✓	✓	✓	
[11]	Case-based	✓	✓	✓	✓		
[14]	Collaborative		✓	✓	✓		
[5]	Hybrid			✓			
[15]	Content-based		✓	✓	✓		
[16]	Content-based			✓		✓	
Our EPR	Hybrid			✓	✓	✓	✓

in this study prevents statistically meaningful computation of such measures. Nevertheless, the observed user feedback suggests that the notifications were effective in supporting users' energy management and optimization efforts. These findings should be interpreted cautiously, as they are based on a small sample size and short-term deployment. A long-term evaluation involving a larger and more diverse user population is required to rigorously assess the system's effectiveness and its sustained impact on energy-saving behavior.

E. Comparative Analysis

Table VII presents a comparative analysis between existing energy recommendation systems and the proposed Energy Personalized Recommender (EPR). The comparison highlights key characteristics such as recommender type, use of price data, level of user control, and supported functionalities including peak-time optimization, anomaly detection, and phantom load detection.

Existing systems, including REHAB-C [5] and other hybrid or single-paradigm approaches, typically focus on a subset of energy management functionalities. For instance, some systems emphasize price-aware scheduling or peak-time reduction, while others address anomaly detection or provide user feedback. However, these approaches do not simultaneously integrate 1) appliance-level monitoring, 2) real-time adaptive recommendations, 3) active user interaction, and 4) multi-objective optimization within a unified framework. As a result, they remain limited either in personalization, responsiveness, or scope of optimization.

The proposed EPR integrates four complementary modules: energy goal tracking, peak-time shifting, phantom load detection, and baseline forecasting. This integration is intended to support multiple energy-management objectives within one system; its comparative effect on energy savings remains to be evaluated.

Eirinaki et al. [4] report energy reductions of 2–17%, while REHAB-C [5] reports reductions of up to 50%. In PowerEye, consumption decreased by 0.75% and 5.28% for two participants but increased by 27.62% for the third, yielding an aggregate increase of 19.74% (Table VI). This preliminary evaluation therefore did not demonstrate an overall energy reduction. Differences in study settings and evaluation methods preclude a direct performance ranking. Table VII describes functional coverage, not superior energy-saving effectiveness.

F. Assumptions and Limitations

The PowerEye project was conducted within the scope of a research-based laboratory prototype, which imposed time, hardware, and deployment constraints that influenced the study outcomes. These constraints limited the duration of data collection and the number of participating users. In particular, the evaluation relied on a small user cohort and a finite dataset, which restricts the statistical generalizability of the reported results.

Hardware resource limitations also constrained real-time processing under higher data volumes and precluded large-scale deployment. Additionally, certain edge cases, such as

irregular appliance usage patterns or atypical household behaviors, were underrepresented, which may affect the robustness of the EPR in more diverse real-world scenarios.

Nevertheless, the primary objective of this study was to validate the functional feasibility and effectiveness of the proposed EPR rather than to provide population-level conclusions. The system is designed as a modular and user-independent framework, and the employed algorithms are computationally scalable. Consequently, the proposed approach is suitable for extension to larger user bases and longer deployment periods.

VII. CONCLUSIONS AND FUTURE WORK

This paper explored PowerEye's EPR for personalized energy-management recommendations. Preliminary results suggest the feasibility of its energy goal, peak time, phantom mode, and baseline algorithms, and participating users reported satisfaction with its notifications. However, the single-day deployment involving three participants did not demonstrate an overall energy reduction or establish sustained user engagement. Longer controlled trials with a larger and more diverse user base are needed to assess prediction accuracy, recommendation effectiveness, and measurable energy savings.

ACKNOWLEDGMENTS

The authors would like to thank the software engineering team members, Maya Emar, Nour Al Akhras, Jody Koujan, Ward Najjar, Raneem Balharith, Sarah Alshagawi, and Renad Almahmoud, for their valuable assistance in developing the system. We also gratefully acknowledge Prince Sultan University for funding this research.

DECLARATIONS

Ethical Approval

This study was approved by the Institutional Review Board of Prince Sultan University (Approval No. PSU-IRB-2025-09-0245).

Consent to Participate

All participants provided informed consent prior to their participation in the study.

Funding

This study was funded by the College of Computers and Information Science at Prince Sultan University.

Competing Interests

The authors declare no competing interests.

Data Availability

The datasets generated and analyzed during the current study are available from the corresponding author upon reasonable request.

Code Availability

The code for this study is publicly available on GitHub at the following link: <https://github.com/rem2718/PowerEye-backend>.

Generative AI Usage Disclosure

The authors used generative AI tools solely for language refinement and readability, with all AI-assisted content reviewed and verified by the authors.

Author Contributions

Conceptualization, R.H. and A.A.; methodology, R.H. and A.A.; software, R.H.; validation, R.H.; formal analysis, R.H.; investigation, R.H.; resources, R.H.; data curation, R.H.; writing—original draft preparation, R.H.; writing—review and editing, A.A., R.H., and S.A.M.; visualization, R.H.; supervision, A.A., S.A.M., and L.A.; project administration, S.A.M., and L.A.; funding acquisition, S.A.M. All authors have read and agreed to the published version of the manuscript.

REFERENCES

- [1] Mahapatra B, Nayyar A (2019) Home energy management system (hems): Concept, architecture, infrastructure, challenges and energy management schemes. *Energy Systems* <https://doi.org/10.1007/s12667-019-00364-w>, URL <https://link.springer.com/article/10.1007/s12667-019-00364-w>
- [2] Himeur Y, Alsalemi A, Al-Kababji A, et al (2021) A survey of recommender systems for energy efficiency in buildings: Principles, challenges and prospects. *Information Fusion* 72:1–21. <https://doi.org/10.1016/j.inffus.2021.02.002>, URL <https://www.sciencedirect.com/science/article/pii/S1566253521000178>
- [3] Sarmas E, Marinakis V, Doukas H (2025) Meta-Learning Approaches for Assessing Energy Efficiency Investments in Buildings, Springer Nature Switzerland, Cham, pp 151–176. https://doi.org/10.1007/978-3-031-85209-1_6, URL https://doi.org/10.1007/978-3-031-85209-1_6
- [4] Eirinaki M, Varlamis I, Dahihande J, et al (2022) Real-time recommendations for energy-efficient appliance usage in households. *Frontiers in Big Data* 5. <https://doi.org/10.3389/fdata.2022.972206>, URL <https://www.frontiersin.org/journals/big-data/articles/10.3389/fdata.2022.972206>
- [5] Sardianos C, Varlamis I, Dimitrakopoulos G, et al (2020) Rehabc: Recommendations for energy habits change. *Future Generation Computer Systems* 112:394–407. <https://doi.org/10.1016/j.future.2020.05.041>, URL <https://www.sciencedirect.com/science/article/pii/S0167739X19317327>
- [6] Badr AA, Saafan MM, Abdelsalam MM, et al (2023) Novel variants of grasshopper optimization algorithm to solve numerical problems and demand side management in smart grids. *Artificial Intelligence Review* 56:10679–10732. <https://doi.org/10.1007/s10462-023-10431-5>, URL <https://doi.org/10.1007/s10462-023-10431-5>
- [7] AbdElkader AG, ZainEldin H, Saafan MM (2024) Optimizing wind power forecasting with rnn-lstm models through grid search cross-validation. *Sustainable Computing: Informatics and Systems* 45:101054. <https://doi.org/10.1016/j.suscom.2024.101054>, URL <https://doi.org/10.1016/j.suscom.2024.101054>
- [8] Lops P, de Gemmis M, Semeraro G (2011) Content-based recommender systems: State of the art and trends. In: Ricci F, Rokach L, Shapira B, et al (eds) *Recommender Systems Handbook*. Springer, Boston MA, p 73–105
- [9] Herlocker JL, Konstan JA, Terveen LG, et al (2004) Evaluating collaborative filtering recommender systems. *ACM Trans Inf Syst* 22(1):5—53. <https://doi.org/10.1145/963770.963772>, URL <https://doi.org/10.1145/963770.963772>
- [10] Jiménez-Bravo DM, Pérez-Marcos J, H. De la Iglesia D, et al (2019) Multi-agent recommendation system for electrical energy optimization and cost saving in smart homes. *Energies* 12(7). <https://doi.org/10.3390/en12071317>, URL <https://www.mdpi.com/1996-1073/12/7/1317>
- [11] González-Briones A, Prieto J, De La Prieta F, et al (2018) Energy optimization using a case-based reasoning strategy. *Sensors* 18(3). <https://doi.org/10.3390/s18030865>, URL <https://www.mdpi.com/1424-8220/18/3/865>

- [12] Villegas NM, Sánchez C, Díaz-Cely J, et al (2018) Characterizing context-aware recommender systems: A systematic literature review. *Knowledge-Based Systems* 140:173–200. <https://doi.org/10.1016/j.knosys.2017.11.003>, URL <https://www.sciencedirect.com/science/article/pii/S0950705117305075>
- [13] Wooldridge M, Jennings NR, Kinny D (2000) The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems* 3(3):285–312. <https://doi.org/10.1023/A:1010071910869>, URL <https://doi.org/10.1023/A:1010071910869>
- [14] Zhang Y, Meng K, Kong W, et al (2019) Collaborative filtering-based electricity plan recommender system. *IEEE Transactions on Industrial Informatics* 15(3):1393–1404. <https://doi.org/10.1109/TII.2018.2856842>
- [15] Zhang Y, Meng K, Xu D, et al (2016) Recommending electricity plans: A data-driven method. In: 2016 IEEE International Conference on Smart Grid Communications (SmartGridComm), pp 668–673. <https://doi.org/10.1109/SmartGridComm.2016.7778838>
- [16] Rodríguez-Cuenca F, Sánchez-Úbeda EF, Portela J, et al (2023) Probability-density-based energy-saving recommendations for household refrigerating appliances. *Engineering Proceedings* 39(1). <https://doi.org/10.3390/engproc2023039043>, URL <https://www.mdpi.com/2673-4591/39/1/43>
- [17] MongoDB (2024) MongoDB atlas: Multi-cloud developer data platform. Available online: <https://www.mongodb.com/atlas> (accessed on 1 March 2025), URL <https://www.mongodb.com/atlas>
- [18] Meross (2024) Meross. Available online: <https://www.meross.com> (accessed on 1 March 2025), URL <https://www.meross.com>
- [19] Geniola A (2020) Merossiot api documentation. Available online: <https://albertogeniola.github.io/MerossIot> (accessed on 1 March 2025), URL <https://albertogeniola.github.io/MerossIot>
- [20] Pallets Projects (2024) Flask, a lightweight wsgi web application framework. Available online: <https://flask.palletsprojects.com/en/stable> (accessed on 1 March 2025), URL <https://flask.palletsprojects.com/en/stable>
- [21] Google (2024) Firebase cloud messaging (fcm). Available online: <https://firebase.google.com/docs/cloud-messaging> (accessed on 1 March 2025), URL <https://firebase.google.com/docs/cloud-messaging>
- [22] Facebook (2024) React native. Available online: <https://reactnative.dev> (accessed on 1 March 2025), URL <https://reactnative.dev>
- [23] Benetti G, Caprino D, Della Vedova ML, et al (2016) Electric load management approaches for peak load reduction: A systematic literature review and state of the art. *Sustainable Cities and Society* 20:124–141. <https://doi.org/10.1016/j.scs.2015.05.002>, URL <https://www.sciencedirect.com/science/article/pii/S2210670715000529>
- [24] Dawson E, Potter J, Meier S (2012) Are phantom loads haunting your energy bill? Available online: http://pages.uoregon.edu/hof/W09HOF/19Phantom_ppr.pdf (accessed on 1 March 2025), URL http://pages.uoregon.edu/hof/W09HOF/19Phantom_ppr.pdf
- [25] Saudi Electricity Company (2024a) Consumption tariffs. Available online: <https://www.se.com.sa/en/OurServices/ColumnC/Bills-and-Consumption/ConsumptionTariffs> (accessed on 1 March 2025), URL <https://www.se.com.sa/en/OurServices/ColumnC/Bills-and-Consumption/ConsumptionTariffs>
- [26] Saudi Electricity Company (2024b) Consumption tariffs, consumption rationalization. Available online: <https://www.se.com.sa/en/Electricity-Information/Consumption-Rationalization/Industrial-Sector> (accessed on 1 March 2025), URL <https://www.se.com.sa/en/Electricity-Information/Consumption-Rationalization/Industrial-Sector>
- [27] Wu J (2012) *Advances in K-means Clustering: A Data Mining Thinking*, 1st edn. Springer Theses, Springer Berlin, Heidelberg, <https://doi.org/10.1007/978-3-642-29807-3>, URL <https://doi.org/10.1007/978-3-642-29807-3>, springer-Verlag Berlin Heidelberg
- [28] Bishop CM (2006) *Pattern Recognition and Machine Learning*. Springer
- [29] Hejazi R (2024) Energy personalized recommender kaggle notebook. Available online: <https://www.kaggle.com/code/reemhejazi/energy-personalized-recommender> (accessed on 1 March 2025), URL <https://www.kaggle.com/code/reemhejazi/energy-personalized-recommender>
- [30] Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, New York, NY, USA, KDD '16, p 785–794. <https://doi.org/10.1145/2939672.2939785>, URL <https://doi.org/10.1145/2939672.2939785>
- [31] Jain A (2024) Xgboost parameters tuning: A complete guide with python codes. Available online: <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python> (accessed on 1 March 2025), URL <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python>
- [32] ARM Software (2020) Mango: A parallel hyperparameter tuning library. Available online: <https://github.com/ARM-software/mango> (accessed on 1 March 2025), URL <https://github.com/ARM-software/mango>
- [33] Zhang J, Zhang H, Ding S, et al (2021) Power consumption predicting and anomaly detection based on transformer and k-means. *Frontiers in Energy Research* 9. <https://doi.org/10.3389/fenrg.2021.779587>, URL <https://www.frontiersin.org/journals/energy-research/articles/10.3389/fenrg.2021.779587>
- [34] Rousseeuw PJ (1987) Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* 20:53–65. [https://doi.org/10.1016/0377-0427\(87\)90125-7](https://doi.org/10.1016/0377-0427(87)90125-7), URL <https://www.sciencedirect.com/science/article/pii/0377042787901257>

APPENDIX A

In this research, home appliances were categorized into three types based on their common energy patterns: Shiftable, Phantom, and None. Shiftable appliances are those whose usage can be adjusted or shifted by residents. Phantom appliances are those that consume power even when they appear to be off (Phantom Mode). The None category includes appliances that do not fit into either of the other two categories. It is important to note that all phantom appliances are considered shiftable, but not all shiftable appliances are phantom. The appliance types are demonstrated in Table VIII.

TABLE VIII. APPLIANCE TYPES

Type	Energy Type	Appliances
Coolers	None	Fridge, Freezer, Mini Fridge, Water Cooler, Fan
Lighting	None	Table Lamp, Desk Lamp, Floor Lamp, Mirror Light
Heaters	None	Microwave, Toaster, Stove, Kettle
Cookers/Makers	None	Coffee Maker, Popcorn Maker, Air Fryer, Slow Cooker
Blenders	None	Blender, Food Processor
No Category	None	Clock/Alexa, Hair Dryer, Camera
Washing Machines	Shiftable	Washer, Dishwasher, Clothes Dryer
No Category	Shiftable	Iron, Vacuum Cleaner, Air Purifier
Games	Phantom	PlayStation, Wii, Xbox
Displays	Phantom	PC, Monitor, Projector, TV
Audio Output	Phantom	Speaker, Radio
No Category	Phantom	Printer, Charger, Receiver, Sports Machine