International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 6, Nov-Dec-2026, ISSN (Online): 2395-566X

Smart Electricity Consumption Analysis and Short-Term Prediction System Using Java and MySQL

Sahil Shashikant Mahabdi[1] , Dr. Deepak Singh[2]
M.Sc. Computer Science
Suryadatta College Of Management Information Research & Technology, Pune, India Research Guide: Dr. Deepak Singh

Abstract- Household electricity use rises and falls with daily routines, the appliances in use, who is at home, the weather, and much else. Without a clear record of readings over time, it is hard for a household to notice when its consumption has changed or to guess what it will need next. This paper proposes a lightweight Smart Electricity Consumption Analysis and Short-Term Prediction System built with Core Java and MySQL. The application stores timestamped readings, works out daily and monthly consumption, and presents usage summaries. It flags unusually high readings with a simple historical-threshold rule and estimates near-term consumption with a moving-average baseline. JDBC links the Java code to the MySQL database, and Java Swing provides the desktop interface. The forecasting component is compared with a simple persistence baseline using Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). Alert quality is judged on labelled test cases using precision, recall, and F1-score. Household load forecasting and appliance-level anomaly detection have already been studied with statistical and machine-learning methods, so this work does not claim to invent electricity forecasting. Its contribution is a low-cost, explainable implementation, together with an empirical evaluation of consumption summaries, simple prediction, and easy-to-understand alerts inside a Java–MySQL application. Results are reported on a synthetic dataset generated for prototype validation; evaluation on real household data is still needed.

Keywords: Electricity consumption, household energy, Java, MySQL, JDBC, moving average, anomaly detection, energy monitoring, short-term prediction.

I. INTRODUCTION

Electricity underpins household life, from study and communication to work. How much a home uses changes constantly, because appliances run at different hours and on different days. A monthly bill gives one total, but it says nothing about when usage climbed or which stretches of time added the most. A well-organised record of readings lets people see their own history and make better choices about how they use energy.

Smart meters and digital monitoring tools now make it possible to collect readings at regular intervals. Researchers have used this kind of data for household demand forecasting, time-series clustering, and anomaly detection. Recent work has looked at predicting household consumption by combining datasets and forecasting models [1], and at using time-series clustering to forecast household demand [2]. A review of appliance anomaly detection also describes statistical, machine-learning, and physical approaches [3].

This paper is about a small, accessible system that a student can realistically build within an M.Sc. project timeline. Java handles the application logic and MySQL handles structured storage. Readings can be typed in by hand or imported from a meter-export file, so no physical smart meter has to be developed. The analysis is kept deliberately transparent, so a user can see why a reading was flagged and how a forecast was calculated.

1. Research Problem And Question

Most households see their electricity use only through periodic bills. When readings are not stored and compared in a consistent way, a sudden rise can go unnoticed for weeks. The forecasting methods described in the literature often need larger datasets, specialised modelling tools, or extra context variables, which puts them out of reach for a small home application. What such an application needs is a practical balance between ease of implementation, outputs people can understand, and performance that can be measured. The study therefore asks: how can a lightweight Java and MySQL application organise household electricity readings, explain changes in consumption, identify unusually high usage, and give a basic short-term estimate using transparent methods?

The objectives are: (1) to design a Java-based application for recording and managing household electricity readings, (2) to store readings and related household or meter details in a structured MySQL database, (3) to calculate daily, weekly, and monthly consumption summaries, (4) to identify unusually high consumption with a documented historical-

© 2026 IJSRET

threshold rule, (5) to estimate short-term consumption with a moving-average forecasting baseline, and

(6) to evaluate the system with forecast-error and anomaly-detection metrics.

The research questions are: (RQ1) how accurately can a moving-average baseline estimate short-term household electricity consumption; (RQ2) can a historical-threshold rule pick out unusually high readings in a labelled test dataset; (RQ3) how does the system perform, in processing time and data retrieval, on a small household dataset; and (RQ4) can the application show consumption changes and alerts in a form a non-technical user can follow?

II. LITERATURE REVIEW

1. Household electricity prediction:
Gaikwad et al. (2024) studied household consumption prediction by combining questionnaire and monthly-consumption datasets, and used feature engineering, data-quality assessment, ensemble prediction, and hybrid modelling [1]. Their results show that prediction is shaped both by changing consumption patterns and by what information the input data actually contains. That is a good reason to document the dataset and its features carefully.

2. Time-series forecasting with smart-meter data:
A 2023 study in Energy Reports looked at time-series clustering and forecasting of household electricity demand from smart-meter data [2]. It compared several forecasting approaches and found that clustering-based methods could lower forecast errors in the setting studied. The lesson here is that a simple historical average should be treated as a baseline, not assumed to be the best option.

3. Appliance Anomaly Detection:
Rauf and Adekoya (2023) reviewed research on anomaly detection and knowledge extraction for household appliances [3]. They grouped the methods into machine-learning, statistical, and physical approaches, and discussed problems such as imbalanced datasets, scarce labelled anomalies, implementation cost, and privacy. The review supports the idea of a transparent threshold-based alert, but it also makes clear that anomaly detection is a well-established field.

4. Wider Residential Demand Estimation:
Peplinski et al. (2024) proposed a machine-learning framework that estimates residential electricity demand from smart-meter data combined with climate, building, and socioeconomic information [4]. It shows how much contextual variables can add, and it contrasts with a smaller system that relies mainly on past readings.

5. Research gap and contribution:
The literature already covers household forecasting, clustering, and anomaly detection, so it would be wrong to call these ideas new. A narrower opportunity lies in building and evaluating an explainable, low-complexity workflow in a Java–MySQL desktop application. That workflow brings together structured reading management, consumption summaries, a transparent high-usage rule, and a moving-average forecast, all tested on one documented dataset. Unless the experiments show a stronger methodological gain, the contribution should be described as one of implementation and evaluation. The proposed contribution is a reproducible Java–MySQL prototype, a controlled comparison of a persistence baseline and a moving-average baseline, and an evaluation of transparent high-usage alerts. The novelty of this combination still needs to be checked through a broader literature and patent search before journal submission.

6. Research Methodology
The study follows a design-and-evaluation approach. The application and database are designed first. Electricity readings are then gathered from a documented public dataset or from a consent-based household recording exercise, cleaned, and stored in MySQL. Next, the analysis and prediction rules are implemented in Java. Finally, the outputs are compared against baseline methods on a time-ordered test set.

7.Data source and collection:
The preferred choice is a public household electricity dataset with timestamped readings, a clearly stated licence, and a data dictionary. One candidate is the household dataset used in published consumption-prediction research [1]. Another option is to collect readings from a consenting household, or to use readings exported from a meter, as long as personal information is removed and permission is obtained. The final paper must name the exact dataset and give its period, sampling interval, number of records, units, missing values, and any preprocessing. For prototype validation in this version, a synthetic dataset was used (see Section IV-D).

8. Data fields:
Table 1 lists the suggested minimum fields for electricity readings.

Field	Description	Example / unit
reading_id	Unique reading identifier	Integer
meter_id	Household meter reference	MTR-001
reading_time	Date and time of reading	2026-01-01 08:00

© 2026 IJSRET

Field	Description	Example / unit
cumulative_kwh	Cumulative meter reading	1250.6 kWh
interval_kwh	Consumption during the interval	0.8 kWh
source_type	Manual / imported / dataset	Imported

Table 1. Suggested minimum fields for electricity readings.*Source: proposed data design for the application.*

9. Data Preprocessing:

- Check timestamps and sort the readings in chronological order.
- Look for duplicate timestamps and duplicate records.
- Look for missing, negative, or physically implausible values, and document any removal or correction.
- If the meter reports cumulative kWh, work out interval consumption by subtracting consecutive readings. Flag negative differences for review, since they may point to a meter reset or a data error.
- Aggregate interval readings into daily totals for daily analysis and forecasting.

10. Synthetic Dataset Used For Prototype Validation:

To validate the pipeline before real data is available, a synthetic daily-consumption series was generated with a fixed random seed (42). It spans 120 days. Weekday consumption is centred on 7.5 kWh and weekend consumption on 9.5 kWh, with Gaussian noise (standard deviation 0.7 kWh). Six high-usage events were injected on days 20, 55, 70, 91, 103 and 112 by multiplying that day's value by a random factor between 1.6 and 1.9; these serve as ground-truth labels. The series was split chronologically: days 1–84 for development and days 85–120 (36 days) for testing, which places three injected events in the test period. The generator and evaluation code are provided as a reproducible Python script. Because the data are simulated, the results below demonstrate that the pipeline works; they do not show how the methods perform on real households.

III. PROPOSED SYSTEM DESIGN

1. **System architecture:**

The system uses a simple layered design. The user enters or imports readings through the Java Swing interface, and the Java application validates the input and sends database operations through JDBC. MySQL stores meter details, readings, alerts, and forecast records. Java service classes calculate consumption summaries, find high-usage days, and generate forecasts, while the interface shows tables, charts, and explanatory messages.

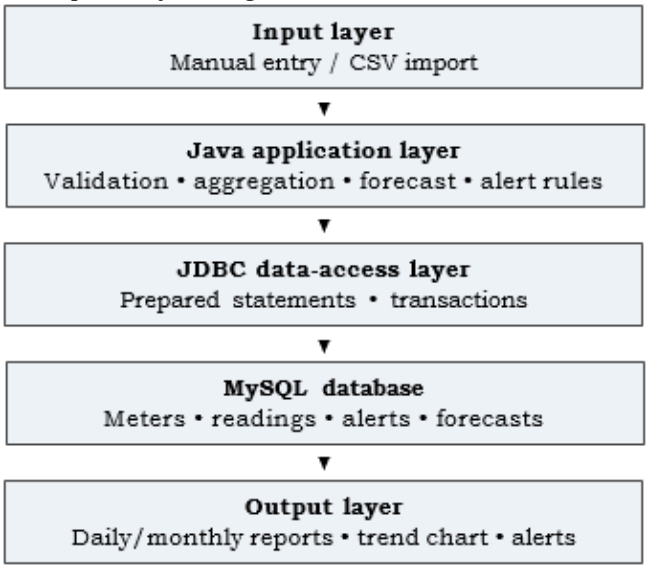

Figure 1. Proposed logical architecture of the Java–MySQL application.

2. Main modules:

- User and meter management: register a household meter and keep its basic details.
- Reading management: add, import, validate, edit, and view electricity readings.
- Consumption analysis: calculate daily, weekly, and monthly totals and compare periods.
- High-usage alert: compare a daily total with a historical reference value and show an explanation.
- Short-term prediction: calculate a simple moving-average forecast for the next period.
- Reports: display reading history, consumption summaries, alerts, and forecast comparisons.

3. Forecasting Method:

As a first implementation, a k-day moving average serves as a transparent forecasting baseline. If C(t) is the daily consumption on day t, the forecast for the next day is:

$$F(t+1) = [\, C(t) + C(t-1) + \ldots + C(t-k+1) \,] / k$$

The project starts with k = 7, provided at least seven valid daily observations are available. The value of k should be chosen using training data only and then tested on later, unseen dates. A persistence baseline, F(t+1) = C(t), is also implemented for comparison, so the moving average is never presented as an improvement without evidence.

© 2026 IJSRET

4. High-Usage Alert Rule:

The alert rule compares the current day's consumption with the average of a preceding reference window. Let μ be the mean daily consumption over the previous n valid days and σ their standard deviation. A candidate alert is raised when the current value exceeds $\mu + \lambda\sigma$, where λ is a configurable sensitivity parameter. The current day must never be part of its own reference window. The alert is worded as "unusually high usage," not as proof of a fault, theft, or appliance malfunction.

A. **Illustrative pseudocode:**

```
INPUT: daily consumption values C, window size k, multiplier λ
1. Sort daily records by date.
2. Validate records; mark missing or invalid days.
3. For each eligible date t, compute forecast F(t) from the previous k days only.
4. Compute mean μ and standard deviation σ from the reference window.
5. If C(t) > μ + λσ, create a high-usage alert.
6. Save forecasts and alerts with timestamps and rule parameters.
7. After the evaluation period, compare forecasts with actual values.
```

5. Database design:

The relational tables in Table 2 are proposed. Primary and foreign keys should be enforced, and prepared statements should be used for every database operation that takes user input.

Table 2. Proposed relational database schema

.

Table	Important columns
meter	meter_id (PK), household_label, meter_type, created_at
reading	reading_id (PK), meter_id (FK), reading_time, cumulative_kwh, interval_kwh, source_type
daily_summary	summary_id (PK), meter_id (FK), summary_date, total_kwh
alert	alert_id (PK), meter_id (FK), alert_date, observed_kwh, threshold_kwh, explanation
forecast	forecast_id (PK), meter_id (FK), forecast_date, predicted_kwh, method, window_size

Source: proposed database design for the application.

6. Evaluation Plan

1. **Experimental setup:**

The dataset is split chronologically into training and test periods. Random shuffling should be avoided for time-series forecasting, because it lets information from later dates leak into predictions for earlier ones. The earliest 70% of dates are used for development and the latest 30% for testing. In the synthetic validation this gives 84 training days and 36 test days.

2. **Forecasting metrics:**

- MAE: the average absolute difference between predicted and actual consumption.
- RMSE: the square root of the average squared error, which gives larger errors more weight.
- MAPE: the average absolute percentage error; days with zero actual consumption must be avoided or handled carefully.

7. Alert Evaluation:

A labelled test set is built in which high-usage events are identified by a documented rule or by independently reviewed examples. Precision, recall, and F1-score are then calculated. If reliable labels cannot be obtained, the paper should report descriptive counts and case reviews rather than claim detection accuracy.

8. Performance And Usability Checks:

- Measure the time needed to import and process a defined number of readings.
- Measure database retrieval time for daily and monthly summaries.
- Test invalid entries, duplicate timestamps, missing readings, and meter resets.
- Ask a small group of volunteers to complete defined tasks and give feedback, but only after obtaining proper consent.

9. Analysis And Results

The results below come from a synthetic dataset (120 days, seed 42) used to validate the pipeline; evaluation on real household data is future work. They show that the forecasting and alert components run correctly end to end, but they should not be read as evidence of performance in real homes. The application's database, runtime and usability measurements (RQ3 and RQ4) have not yet been carried out.

© 2026 IJSRET

Table 3. Forecasting performance on the 36-day test period (synthetic data).

Method	MAE (kWh)	RMSE (kWh)	MAPE (%)
Persistence baseline	1.921	2.755	21.23
7-day moving average	1.523	2.216	16.82

Source: results on a synthetic dataset (120 days, seed 42) generated for prototype validation; not real household measurements.

Table 4. High-usage alert evaluation (n = 14, λ = 2.0; 36-day test period, synthetic data).

Measure	Value
True positives	2
False positives	0
False negatives	1
Precision / Recall / F1-score	1.00 / 0.67 / 0.80

Source: results on a synthetic dataset (120 days, seed 42) generated for prototype validation; not real household measurements.

On the test period the 7-day moving average gave lower error than the persistence baseline on all three metrics: MAE fell from 1.921 to 1.523 kWh (about 21% lower), RMSE from 2.755 to

3. Kwh, And Mape From 21.23% To 16.82%

Excluding the three spike days, MAE was

1.140 kWh for the moving average against 1.632 kWh for persistence. Choosing k on training data only, among 3, 5, 7, 10 and 14, also selected k = 7. The alert rule raised no false alarms and detected two of the three injected events in the test period, missing one. Reproducing these numbers requires only the accompanying script and its fixed seed.

4. Findings And Discussion

In this synthetic experiment the moving average outperformed persistence, which is expected: persistence repeats yesterday's value, so it inherits the weekday/weekend swing and any one-day spike, while averaging seven days smooths both. The simulated data contain a regular weekly pattern and independent noise, which favour an averaging method, so the size of the gain should not be expected to carry over to real households with occupancy changes, seasonal effects, or holidays. The threshold alert caught two of three spikes with no false alarms, but its recall of 0.67 comes from only three labelled events, so the estimate is very uncertain. A threshold alert can spot a statistical deviation, but it cannot tell what physically caused the extra consumption.

The discussion should also separate software functionality from research effectiveness. Showing that Java can store and display readings is a software result. Showing, through a documented experiment on real data, that a chosen method gives useful forecasts or alerts is the research result. The latter remains to be done.

IV. CONCLUSION

This paper has proposed a Smart Electricity Consumption Analysis and Short-Term Prediction System built with Core Java and MySQL. It combines structured reading management, daily and monthly summaries, a transparent high-usage alert rule, and a moving-average forecasting baseline. A prototype validation on synthetic data showed the moving average improving on a persistence baseline and the alert rule detecting two of three injected events without false alarms. The design should be feasible for an M.Sc. Computer Science project and needs no specialised hardware when historical readings are available. Existing studies show that household electricity forecasting and anomaly detection are active research areas, so the work is positioned as a lightweight implementation and evaluation, not as a claim that these tasks are new. How useful the system is in practice must still be settled by testing on a documented real dataset.

1. Limitations And Future Scope

1. Limitations:

- Reported results come from synthetic data and do not establish real-world performance.
- Forecast quality depends on how much history there is, and on its quality and sampling frequency.
- A moving average may miss holidays, weather changes, appliance-level behaviour, or sudden lifestyle changes.
- A high-usage alert shows a deviation from a reference pattern; it does not diagnose the cause.
- Manual readings may be less frequent and less precise than smart-meter readings.
- The desktop application does not control appliances or monitor the utility grid in real time.

2. Future scope:

- Evaluate on a real public dataset or consented household data, and on multiple households.
- Compare the moving average with other statistical forecasting methods [5] and with deep-learning models such as bi-directional LSTM [6].

© 2026 IJSRET

- Add weather, weekday, weekend, and holiday features where reliable data exists.
- Support optional appliance-level readings for users who have suitable measurement data.
- Build a web or mobile interface while keeping the Java–MySQL backend.
- Study privacy-preserving storage and access control for household consumption records.

REFERENCES

1. S. R. Gaikwad, R. Harikrishnan, S. M. Muyeen, and K. Kotecha, "Household electricity consumption prediction using database combinations, ensemble and hybrid modeling techniques," Scientific Reports, vol. 14, art. 22891, 2024. DOI: 10.1038/s41598-024-57550-9.
2. "Time-series clustering and forecasting household electricity demand using smart meter data," Energy Reports, vol. 9, pp. 4111–4121, 2023. DOI: 10.1016/j.egyr.2023.03.042.
3. S. A. A. Rauf and A. F. Adekoya, "Systematic literature review of the techniques for household electrical appliance anomaly detections and knowledge extractions," Journal of Electrical Systems and Information Technology, vol. 10, art. 22, 2023. DOI: 10.1186/s43067-023-00086-1.
4. M. Peplinski, B. Dilkina, M. Chen, S. J. Silva, G. A. Ban-Weiss, and K. T. Sanders, "A machine learning framework to estimate residential electricity demand based on smart meter electricity, climate, building characteristics, and socioeconomic datasets," Applied Energy, vol. 357, art. 122413, 2024. DOI: 10.1016/j.apenergy.2023.122413.
5. I. Moustati, N. Gherabi, and M. Saadi, "Time-Series Forecasting Models for Smart Meters Data: An Empirical Comparison and Analysis," Journal Européen des Systèmes Automatisés, vol. 57, no. 5, pp. 1419–1427, 2024. DOI: 10.18280/jesa.570517.
6. K. Aurangzeb, S. I. Haider, and M. Alhussein, "Individual household load forecasting using bi-directional LSTM network with time-based embedding," Energy Reports, vol. 11, pp. 3963–3975, 2024. DOI: 10.1016/j.egyr.2024.03.028.
7. F. Rodrigues, C. Cardeira, J. M. F. Calado, and R. Melicio, "Short-Term Load Forecasting of Electricity Demand for the Residential Sector Based on Modelling Techniques: A Systematic Review," Energies, vol. 16, no. 10, art. 4098, 2023. DOI: 10.3390/en16104098.
8. P. Ma, S. Cui, M. Chen, S. Zhou, and K. Wang, "Review of Family-Level Short-Term Load Forecasting and Its Application in Household Energy Management System," Energies, vol. 16, no. 15, art. 5809, 2023. DOI: 10.3390/en16155809.
9. A. Mehdipour Pirbazari, M. Farmanbar, A. Chakravorty, and C. Rong, "Short-Term Load Forecasting Using Smart Meter Data: A Generalization Analysis," Processes, vol. 8, no. 4, art. 484, 2020. DOI: 10.3390/pr8040484.
10. S. Haben, S. Arora, G. Giasemidis, M. Voss, and D. Vukadinović Greetham, "Review of Low Voltage Load Forecasting: Methods, Applications, and Recommendations," Applied Energy, vol. 304, art. 117798, 2021. DOI: 10.1016/j.apenergy.2021.117798.
11. A. Ambat and J. Sahoo, "Anomaly Detection and Prediction of Energy Consumption for Smart Homes Using Machine Learning," ETRI Journal, vol. 47, no. 5, pp. 934–945, 2025. DOI: 10.4218/etrij.2023-0155.
12. L. Botman, J. Lago, T. Becker, K. Vanthournout, and B. De Moor, "A Global Probabilistic Approach for Short-Term Forecasting of Individual Households Electricity Consumption," Applied Energy, vol. 382, art. 125168, 2025. DOI: 10.1016/j.apenergy.2024.125168.

© 2026 IJSRET