SolarWise

DYNAMIC AI-DRIVEN ENERGY MANAGEMENT CLOUD SOLUTIONS FOR CONTROL, FORECASTING, OPTIMIZATION, AND ALERTS

By Team Techno Titans

- Manvendra Singh- Grafana, UI/UX Design (IIIT Raichur)
- Pawan Kumar- Database, Frontend (IIIT Raichur)
- Pratham Jain- AI/ML and Data Wrangling (IIIT Raichur)

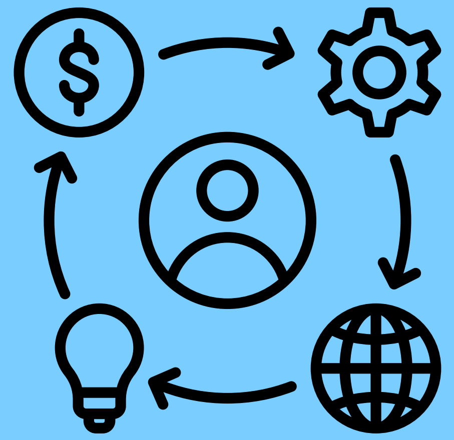
Product Vision

Dynamic AI-Driven Energy Management Cloud Solutions for Control, Forecasting, Optimization, and Alerts

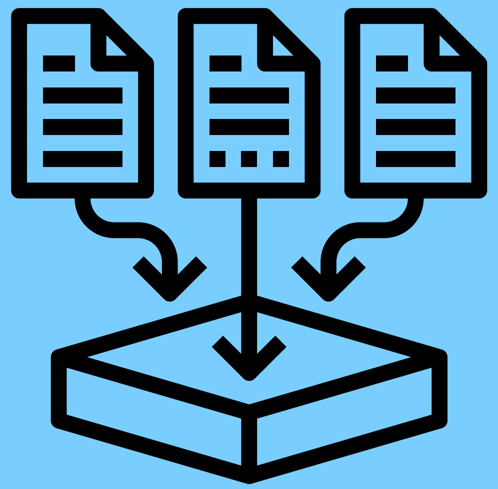
Value Proposition

- Customer Retention
- Revenue Generation
- Data Monetization
- Brand Differentiation
- Energy Market Leadership

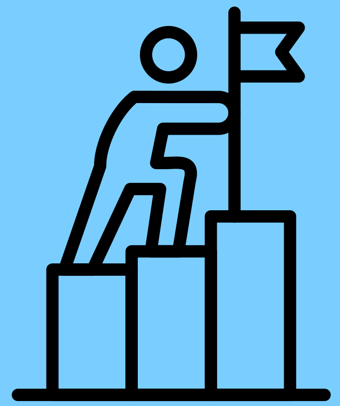
20-30% more savings*



Bring Users to Luminous Ecosystem



Insights from customer energy usage



Set our brand apart from competition

Product-Market Fit

- **Residential and commercial fit**, expanding Luminous target audience.

Market Research

Solar installations are projected to grow by 25% annually, and battery storage adoption by 15% annually

The global IoT-enabled energy management market is projected to grow at a **CAGR of 15.2%, reaching $15.3 billion by 2030 ($3 billion)**

Offering AI-based optimization **differentiates Luminous as an innovator** in the renewable energy market as **71% of energy consumers** prefer providers offering digital tools and AI-driven insights

SolarWise: Aligning with Luminous Goals

Reaching Every Home

- Adapt to households of all sizes and business needs.
- Accessibility for everyone.
- Cloud-Based Seamless web implementation for massive adoption

Promoting Renewable Energy

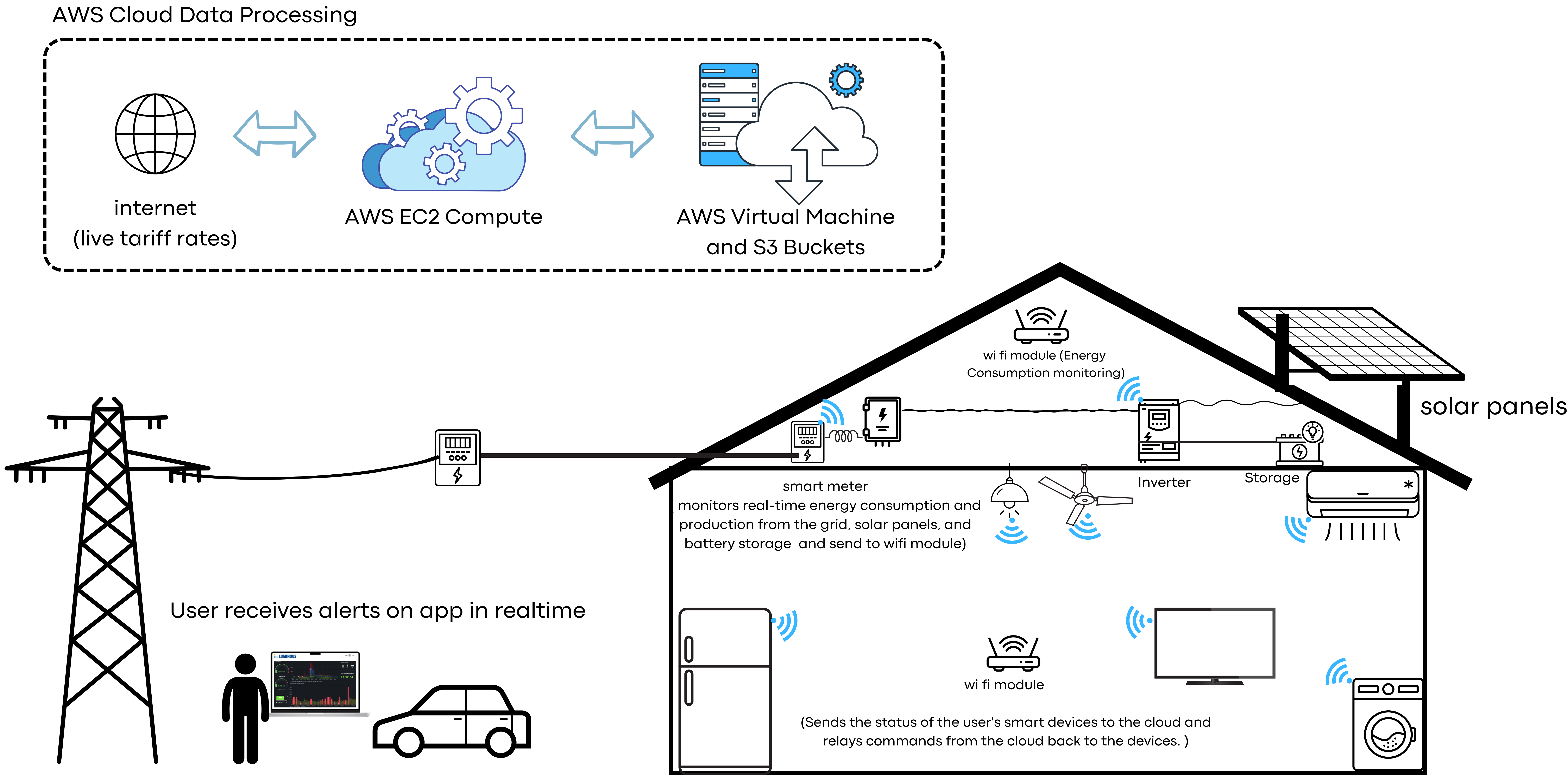
- Provides real-time insights, cost-saving recommendations, and energy optimization
- Enhances user’s awareness on environmental impact

User Flexibility: Users can monitor and manage their energy consumption from anywhere with an internet connection. This remote access eliminates the need for on-site presence, making it convenient for users to stay informed and in control

Predicts Future Tariff Rates	Analysis and Predicts User Consumption	Anomaly Detection for user power usage	Live Solar Power Generation and Usage
Suggestions /Alerts	Finds and calculates Cost at which we can sell energy back to grid	Smart Device Scheduling and Monitering	Savings by Smart Load Balancing

Assumption#1

IoT Architecture for Realtime Energy Consumption Monitoring, Smart Scheduling and Alerts (push notifications)



Assumption#2

We are using the TOU based tariff data generated using Tarrif of Time of Use (ToU) Indian Power System Dataset from Mendley Data which we normalise for commercial rates per unit and then loop to simulate the oncoming tariff rates
PFA refernce"https://data.mendeley.com/datasets/7g9frz6sm8/1"

Assumption#3 for smart scheduling

We assume that the user has some tracked data consumption other than tracked from these IoT device hence,
User Consumption = Sum of IoT devices that he/she wishes to schedule beforehand + some non IoT devices (which is not constant as well)

Assumption#4

Similarly we also loop the cleaned data derived from data provided by Luminous after the first QnA sesion to mimic solar power generation and consumption

We have implemented the models and computed the values beforehand for 7 days to avoid costs of realtime compute on AWS Sagmaker but it mimics fetching these values using graphana from the aggreagted database tables

Assumption#5 NO Garbage Data

All machine learning models developed so far have been trained using the split and augmented data mentioned above. These models serve as a solid starting point for real consumer data, as they have been validated against actual user consumption data from InAnalytics.!!

Aggregated Data Sent to Cloud

	A	B	C	D	E	F	G	H	I
1	Solar energy Gener	consumptionValue	Device_1_Consum	Device_2_Consum	Device_3_Consum	Device_4_Consum	TOU_rates (INR)	Cummulative Ener	Senddate
2	0	0.37	0.103	0.051	0.01	0.184	10	0.37	01-07-2024 00:03
3	0	0.39	0.071	0	0.062	0.058	10	0.76	01-07-2024 00:07
4	0	0.37	0.127	0.005	0.004	0.016	10	1.13	01-07-2024 00:13
5	0	0.38	0.121	0.195	0.231	0.038	10	1.51	01-07-2024 00:19
6	0	0.38	0.005	0.137	0.078	0.146	10	1.89	01-07-2024 00:24
7	0	0.37	0.016	0.033	0.046	0.052	10	2.26	01-07-2024 00:30
8	0	0.37	0.232	0.106	0.082	0.107	10	2.63	01-07-2024 00:35
9	0	0.37	0.233	0.132	0.091	0.037	10	3	01-07-2024 00:40
10	0	0.36	0.01	0.116	0.139	0.251	10	3.36	01-07-2024 00:46
11	0	0.38	0.011	0.205	0.085	0.09	10	3.74	01-07-2024 00:51
12	0	0.37	0.102	0.236	0.132	0.03	10	4.11	01-07-2024 00:56
13	0	0.37	0.066	0.165	0.178	0.101	10	4.48	01-07-2024 01:01
14	0	0.37	0.037	0.026	0.008	0.09	10	4.85	01-07-2024 01:07
15	0	0.37	0.039	0.083	0	0.041	10	5.22	01-07-2024 01:12
16	0	0.37	0.27	0.121	0.049	0.072	10	5.59	01-07-2024 01:17
17	0	0.37	0.046	0.122	0.043	0.192	10	5.96	01-07-2024 01:23

User Consumption Prediction

LSTM

- Generally achieves around 92% accuracy.
- LSTMs capture long-term dependencies in sequential data, beneficial for time series. However, they are prone to overfitting, require more data for training, and have longer training times.

Predictive Rates of TOU Tariff

Linear Regression

- Provides robust predictions with an R-squared value often above 90%.
- As a lightweight model, it requires less computational power and is easier to interpret, allowing for quick insights without sacrificing accuracy compared to more complex models like Random Forest, which may hover around 85% due to overfitting.

Anomaly Detection

Z-Score

Z-Score or IQR methods provide precision rates above 95% and are computationally inexpensive, making them more efficient than Isolation Forest and other methods in literature

Device Scheduling to Minimize Cost ⚠

Mixed-Integer Linear Programming (MILP)

- Can achieve optimal scheduling solutions with guarantees of 100% optimality.
- MILP guarantees optimal solutions and efficiently handles constraints, making it superior for precise scheduling tasks compared to heuristic methods like Genetic Algorithms, which may provide solutions with lower optimality between 85-90%
-

Minimize the total cost of grid usage by:

$$\text{Total Cost} = \sum (\text{Grid Used (kW)} \times \text{Tariff (INR/kWh)})$$

Objective function for balancing grid and solar

Additionally, maximize savings by reducing grid consumption through solar and battery use, and minimize environmental impact by increasing solar usage (trees saved).

$$\text{Minimize: } \sum_{i=1}^{n} \text{Energy consumed by device } i \times \text{Tariff rate during scheduled period}$$

Smart Scheduling Algorithm for IoT devices

Key Constraints:

1. Devices must be scheduled during low tariff periods.
2. High-priority devices should be scheduled first, followed by low-priority ones.
3. Energy consumption should be calculated for each device during the scheduled time.
4. Energy consumption should be updated in the dataframe.

16	Normalize method	zscore
17	Feature selection	True
18	Feature selection method	classic
19	Feature selection estimator	lightgbm
20	Number of features selected	0.200000
21	Fold Generator	TimeSeriesSplit
22	Fold Number	10
23	CPU Jobs	-1
24	Use GPU	False
25	Log Experiment	MlflowLogger
26	Experiment Name	TOU_rates_INR_experiment
27	USI	ae2f

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
gbr	Gradient Boosting Regressor	1.2335	11.3174	1.9288	0.6652	0.1140	0.1063	0.1220
lightgbm	Light Gradient Boosting Machine	1.2908	11.5600	1.9910	0.6540	0.1163	0.1076	0.1600
et	Extra Trees Regressor	1.3632	11.8618	1.9900	0.6393	0.1172	0.1126	0.1390
rf	Random Forest Regressor	1.3503	12.4517	2.0952	0.6174	0.1210	0.1091	0.1520
dt	Decision Tree Regressor	1.3467	12.5071	2.1411	0.6158	0.1233	0.1091	0.0870
ada	AdaBoost Regressor	2.0980	14.0836	3.0453	0.5937	0.1608	0.1446	0.0980
knn	K Neighbors Regressor	2.8617	21.6687	4.1877	0.3355	0.2456	0.2122	0.0900
dummy	Dummy Regressor	5.6815	40.5182	6.3546	-0.1138	0.3826	0.4178	0.0910
llar	Lasso Least Angle Regression	5.7707	41.9349	6.4473	-0.1575	0.3873	0.4265	0.0860
lasso	Lasso Regression	5.7707	41.9349	6.4473	-0.1575	0.3873	0.4265	0.0900
en	Elastic Net	5.7980	42.3777	6.4740	-0.1759	0.3887	0.4289	0.0870
br	Bayesian Ridge	5.9569	45.6863	6.6666	-0.2882	0.3960	0.4387	0.0860
omp	Orthogonal Matching Pursuit	5.9909	46.4417	6.7121	-0.3137	0.3989	0.4433	0.0920
ridge	Ridge Regression	5.9800	46.4574	6.7160	-0.3141	0.3982	0.4404	0.0890
lr	Linear Regression	5.9805	46.4725	6.7169	-0.3146	0.3982	0.4405	0.5800
lar	Least Angle Regression	5.9805	46.4725	6.7169	-0.3146	0.3982	0.4405	0.0880
huber	Huber Regressor	6.1915	51.4564	7.0427	-0.4608	0.4112	0.4579	0.0860
par	Passive Aggressive Regressor	6.8969	73.3879	8.3759	-1.0154	0.4676	0.5004	0.0850

PYCARET

We have trained various shallow and deep models on our toy data with shallow models performing better

but...

on real-life data we had available to us from previous projects we know that these models tend to overfit to data and are unable to capture complexity

10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Remove multicollinearity	True
14	Multicollinearity threshold	0.900000
15	Normalize	True
16	Normalize method	zscore
17	Feature selection	True
18	Feature selection method	classic
19	Feature selection estimator	lightgbm
20	Number of features selected	0.200000
21	Fold Generator	TimeSeriesSplit
22	Fold Number	10
23	CPU Jobs	-1
24	Use GPU	False
25	Log Experiment	MlflowLogger
26	Experiment Name	Solar_energy_Generation_kWh_experiment
27	USI	0fd5

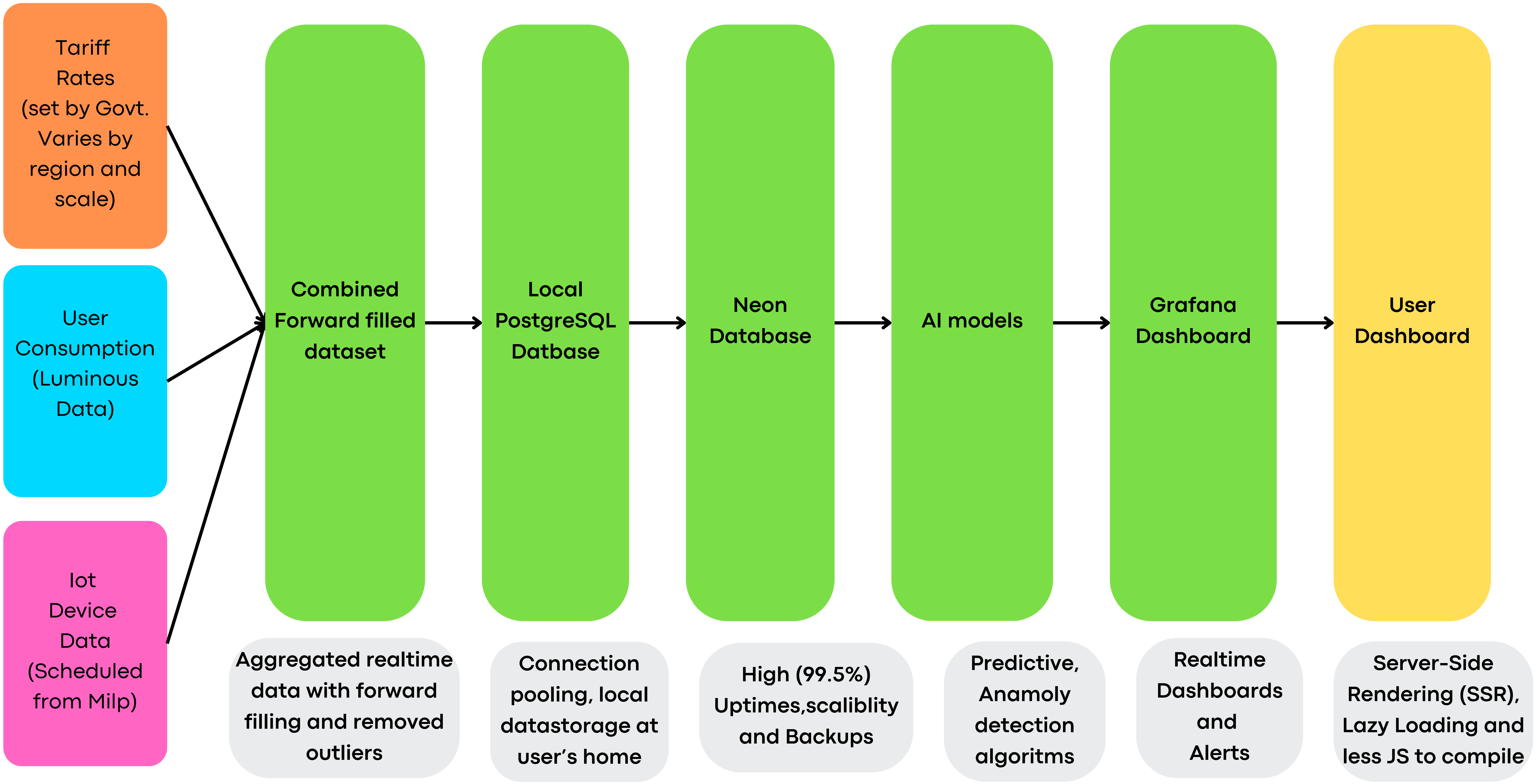
```
2024/10/26 12:30:32 INFO mlflow.tracking.fluent: Experiment with name 'Solar_energy_Generation_kWh_experiment' does not exi
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
dt	Decision Tree Regressor	1.1027	6.2215	1.2971	-1.1294	0.4687	nan	0.0950
et	Extra Trees Regressor	1.0346	5.6096	1.2367	-1.2250	0.4592	nan	0.1340
rf	Random Forest Regressor	1.0731	5.9458	1.2627	-1.3060	0.4584	nan	0.1760
knn	K Neighbors Regressor	1.1118	5.6694	1.3807	-1.3544	0.5204	nan	0.1080
lightgbm	Light Gradient Boosting Machine	1.2387	6.2642	1.4458	-1.4362	0.5449	nan	0.1410
ada	AdaBoost Regressor	1.1345	5.4740	1.3713	-6.7232	0.5463	nan	0.0980
gbr	Gradient Boosting Regressor	1.0648	5.6522	1.2803	-19.0890	0.4704	nan	0.1380
huber	Huber Regressor	1.8303	19.0092	2.0902	-34.3913	0.6269	nan	0.0930
br	Bayesian Ridge	2.1355	18.1406	2.4422	-63.3702	0.7837	nan	0.1060
ridge	Ridge Regression	2.1462	18.1829	2.4562	-64.0387	0.7870	nan	0.0850
lr	Linear Regression	2.1506	18.2746	2.4610	-64.1000	0.7876	nan	0.1070
lar	Least Angle Regression	2.1506	18.2746	2.4610	-64.1005	0.7876	nan	0.0870
dummy	Dummy Regressor	1.1412	1.8787	1.2461	-101.7768	0.6589	nan	0.0860
en	Elastic Net	1.6188	7.3443	1.7382	-102.9597	0.7275	nan	0.0870
lasso	Lasso Regression	1.7429	8.9880	1.8664	-103.4939	0.7598	nan	0.0980
llar	Lasso Least Angle Regression	1.7429	8.9880	1.8664	-103.4939	0.7598	nan	0.0880
omp	Orthogonal Matching Pursuit	1.9890	17.5363	2.3207	-159.3547	0.7281	nan	0.0880
par	Passive Aggressive Regressor	2.1864	13.3521	2.5096	-195.9846	0.8269	nan	0.0910

This can better seen in models trained to predict Solar_energy_consumption for downstream tasks such as scheduling, prediction of savings, etc..

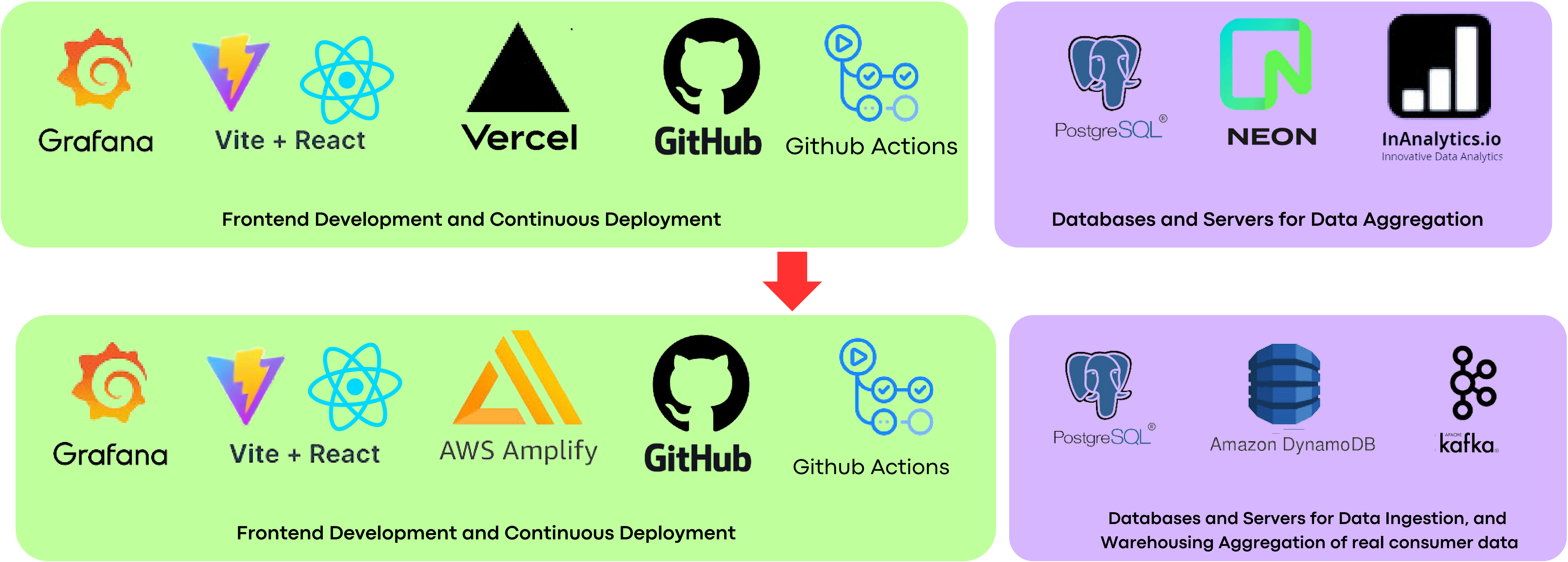
here we can see the Although the Mean Squared Error (MSE) values are reasonable, **the R^2 values are relatively low. or negative R^2 values**, indicating that they perform poorly in capturing the variance in the data. This suggests that while the models might have acceptable error rates (as measured by MSE), they fail to explain the underlying patterns in the data effectively.

Proof of Concept Implementation

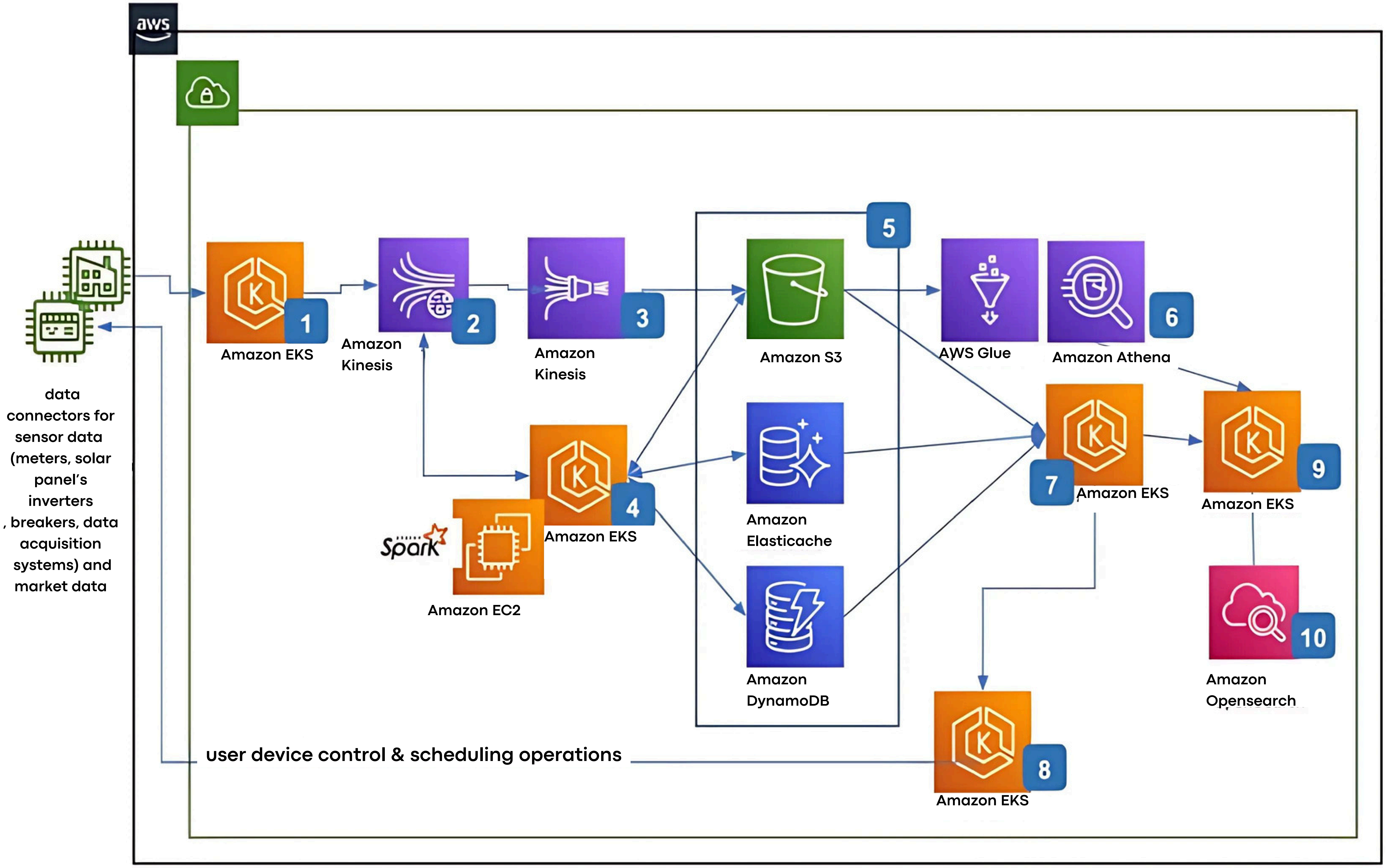


- Vercel → AWS Amplify Hosting
- Local PostgreSQL Database → Amazon Aurora
- GitHub Actions → AWS CodePipeline
- Custom APIs → Custom APIs on Amazon EKS
- Real-Time Data Processing → Amazon Kinesis
- Stream Processing → Amazon EKS for custom stream processing listeners
- Neo4j → Amazon Neptune (if replacing)

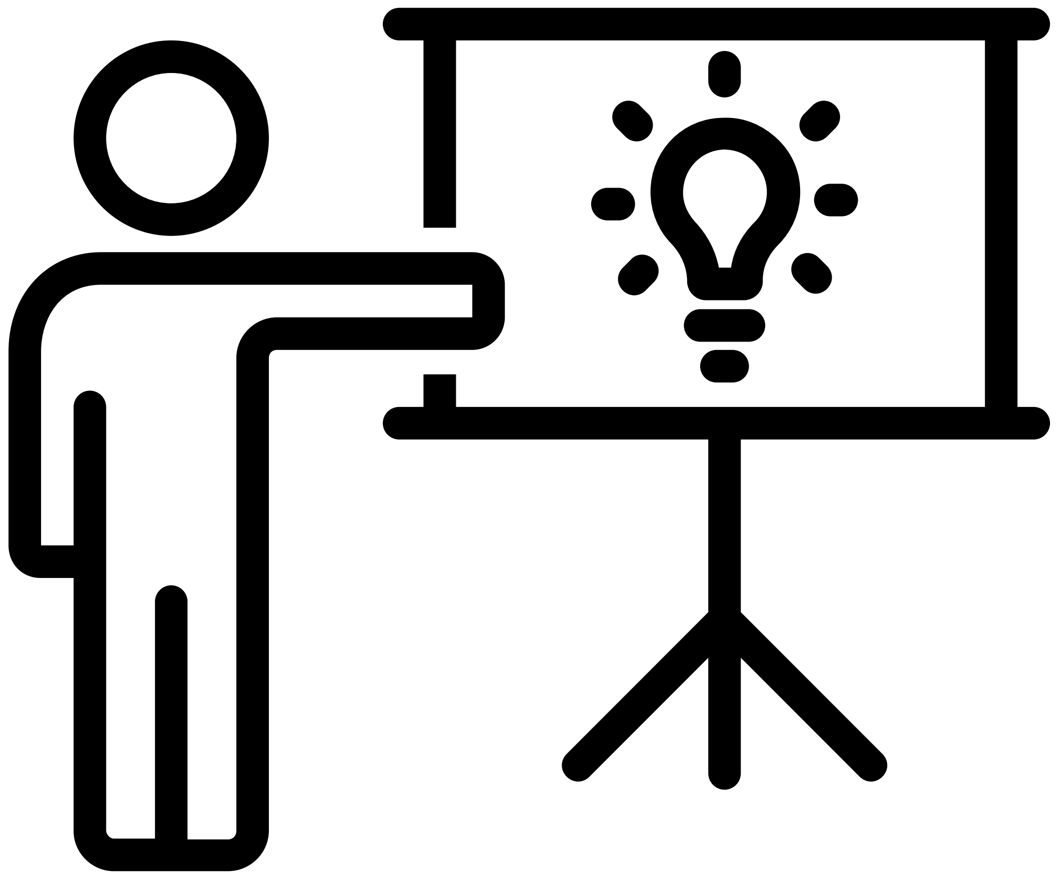
With minimal changes, we can deploy our proof of concept into a horizontally scalable system for real-time and stream processing. enabling seamless data ingestion and processing to meet growing user demands.



Our architecture design for FULL Scale Dynamic AI-Driven Energy Management Cloud Solutions for Control, Forecasting, Optimization, and Alerts



1. Custom-built, protocol-specific **data connectors** for sensor data (meters, solar panel's inverters, breakers data acquisition systems) and market data are deployed on **Amazon EKS.**
2. Incoming data is streamed into **Amazon Kinesis**.
3. **Amazon Kinesis** Firehose provides no-code persistence in Parquet or **JSON** lines format.
4. Custom stream processing listeners deployed using **Amazon EKS** additionally perform value-added **data transformation**, **enrichment**, and **aggregations**.
5. The polyglot storage architecture leverages **S3**, **ElastiCache**, **DynamoDB**, **Aurora**, and **OpenSearch** to cater to various application needs.
6. **Amazon Athena and S3** select APIs provide effective **serverless query** capabilities over data in **S3**.
7. **Custom APIs deployed on Amazon EKS blend data** from various storage systems to provide an integrated feature store used by ML processes.
8. AI and ML tasks for **forecasting, optimization, and control are deployed on EKS**.
9. Advanced **rule-based processing** is used for **near real-time monitoring** with **AWS Athena Queries**.
10. **Elasticsearch-based dashboards** provide real-time insights for user device control operations.



Let’s have a look at our demonstration

WE WANT YOUR

FEEDBACK

Thank you for your time

Full Scale Approach Overview

The system integrates multiple components for AI-powered Energy Management with a focus on Forecasting, Optimization, and Control on AWS:

1. **IoT-Enabled Energy Devices and Market Data Integration:**
    - Intelligent Energy Devices (IED): Meters, inverters, energy storage systems (ESS), and other devices capture real-time energy consumption, solar production and solar battery levels and market data (tariff rates). This data is ingested and transferred using wi-fi modules to the cloud, allowing for monitoring and dynamic control of energy resources.
2. **Cloud-Based Data Processing Platform:**
    - **Data Ingestion**: Incoming data from intelligent devices is streamed into Amazon Kinesis for real-time processing. Custom-built connectors handle data from meters, inverters, breakers, and other acquisition systems.
    - **Stream Processing and Transformation**: Stream listeners, deployed on Amazon EKS (Elastic Kubernetes Service), transform and enrich the data, leveraging Apache Spark for distributed data processing.
    - **Data Storage:** The processed data is stored in a polyglot architecture involving Amazon S3, ElastiCache, DynamoDB, Aurora, and OpenSearch, ensuring efficient storage and retrieval for both real-time and historical analytics.
3. **Machine Learning for Energy Forecasting and Optimization:**
    - **ML Task Deployment**: Machine learning models for energy forecasting, optimization, and control are deployed on Amazon EKS. These models are trained on the processed data, enabling energy management features such as load forecasting and optimization.
    - **Feature Store Integration**: Custom APIs deployed on Amazon EKS consolidate data from multiple sources to form an integrated feature store, allowing ML models to make predictions with high accuracy.

Real-Time Querying and Monitoring:

- **Serverless Queries**: Data stored in Amazon S3 is queried using Amazon Athena for real-time monitoring and reporting. This facilitates timely decision-making without the need for heavy infrastructure.
- **Anomaly Detection**: Advanced rule-based processing monitors the incoming data streams for anomalies, such as unexpected spikes in consumption. If an anomaly is detected, automated workflows trigger alerts to notify the user.

User Alerts and Control:

- **Control Energy Resources**: The system also has provisions to control energy devices dynamically, enabling optimization based on forecasted consumption and tariffs.
- **Notification and Dashboarding**: Elasticsearch-based dashboards display system health and performance metrics to network operation centers (NOC) for proactive management.

-: Important details to note : -

We have implemented the models and computed the values beforehand for 7 days to avoid costs of realtime compute on AWS Sagmaker but it mimics fetching these values using graphana from the aggreagted database tables

We are using the TOU based tariff data generated using Tarrif of Time of Use (ToU) Indian Power System Dataset from Mendley Data which we normalise for commercial rates per unit and then loop to simulate the oncoming tariff rates
PFA refernce"https://data.mendeley.com/datasets/7g9frz6sm8/1"

Similarly we also loop the cleaned data provided by Luminous after the first QnA sesion to mimic solar power generation and consumption

All machine learning models developed so far have been trained using the split and augmented data mentioned above. These models serve as a solid starting point for real consumer data, as they have been validated against actual user consumption data from InAnalytics.!!