

**College of Engineering**

**COMP 491 - Computer Engineering Design**

**Project Proposal**

**ForeWatt**

**Energy Demand Forecasting and Anomaly Detection**

**Fall 2025**

**Participant Information:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **ID** | **Email** | **Phone** |
| Zeynep Öykü Aslan | 79731 | zeynepaslan21@ku.edu.tr | 5313878595 |
| Kaan Altaş | 79855 | [kaltas21@ku.edu.tr](mailto:kaltas21@ku.edu.tr) | 5351008611 |
| Zeliha Paycı | 79683 | [zpayci21@ku.edu.tr](mailto:zpayci21@ku.edu.tr) | 5449008975 |

**Project Advisor:** Prof. Dr. Gözde Gül Şahin

**Abstract**

ForeWatt is a reproducible, end to end platform for hourly electricity demand forecasting, calibrated uncertainty, and actionable diagnostics over horizons from 1 to 24 hours. The project is motivated by the operational need to plan procurement and demand response with trustworthy predictions that remain auditable under real world constraints of limited compute and zero license cost. We ingest open hourly load from EPİAŞ and benchmarking data from PJM together with weather covariates from Open-Meteo, transform them through a medallion pipeline, and expose a consistent feature set for training and serving. The core forecaster is a modern time series model such as N-HiTS, complemented by transparent baselines including Prophet and gradient boosting. Uncertainty is communicated through split conformal prediction intervals that target reliable coverage at the 90 percent nominal level. Anomaly detection combines residual monitoring with IsolationForest and presents level shift, weekly drift, and feature attribution diagnostics to make alerts actionable. The system is delivered as a Docker based stack with FastAPI for serving, MLflow for experiment tracking and model registry, QuestDB for time series storage, and a Streamlit dashboard for interactive visualization. The objectives are to achieve competitive sMAPE and MASE on EPİAŞ subsets, maintain freshness within two hours, keep single horizon latency below 300 milliseconds on CPU, and demonstrate a stretch outcome that converts forecasts into cost reducing schedules for controllable loads. The desired outcome is a production ready, fully documented platform that is easy to rerun and extend.

**Table of contents**

[Section 1 Introduction 4](#_Toc211819638)

[1.1 Concept 4](#_Toc211819639)

[1.2 Objectives 4](#_Toc211819640)

[1.3 Background 5](#_Toc211819641)

[Section 2 S/T Methodology and Associated Work Plan 7](#_Toc211819642)

[2.1 Methodology 7](#_Toc211819643)

[2.2 Work Package Descriptions 16](#_Toc211819656)

[2.3 Demonstration 20](#_Toc211819657)

[2.4 Impact 21](#_Toc211819658)

[2.5 Risk analysis 21](#_Toc211819659)

[2.6 Gantt Chart 22](#_Toc211819660)

[Section 3 Economical and Ethical Issues 22](#_Toc211819661)

[Section 4 References 24](#_Toc211819662)

1. Introduction
   1. Concept

ForeWatt is an end to end, open source energy analytics platform that forecasts Turkey’s electricity demand, detects operational anomalies, and recommends cost reducing actions. The project addresses a timely need for transparent, reliable, and reproducible forecasting at grid scale, turning publicly available market and weather data into actionable intelligence for planners and operators. The core objectives are to deliver calibrated day ahead and short term forecasts at hourly resolution, surface explained anomaly alerts rather than opaque flags, quantify economic impact through an optimization layer, and demonstrate production grade MLOps practices.

The methodology follows a disciplined data and modeling pipeline. Periodic ingestion from EPİAŞ (load and prices), PJM (benchmarking), and Open-Meteo (weather) lands in a bronze-silver-gold medallion architecture with schema and range checks, conservative gap handling, and a strict freshness guard (≤ 2 hours). Feature engineering captures short term momentum and weekly seasonality (lags, rolling statistics, Fourier terms), calendar effects, and weather covariates, with deterministic, versioned transformers to preserve offline/online parity. Forecasting centers on a strong neural model (N-HiTS) complemented by interpretable and tabular baselines (Prophet, CatBoost), with temporal cross validation, Bayesian tuning caps, and lightweight alternatives as time allows. Uncertainty is communicated via split conformal prediction to provide finite sample coverage guarantees per horizon.

Operations and decision support are integrated from the start. An IsolationForest based diagnostic layer produces anomaly alerts with level shift and weekly drift tests plus feature attributions. A linear programming heuristic transforms forecasts into EV charging schedules that respect power and energy by deadline constraints and report cost deltas and feasibility slack. The system is containerized (Docker) and served via FastAPI with a Streamlit dashboard; MLflow tracks experiments, artifacts, and champion, challenger promotion for safe, auditable iteration.

Expected outcomes include day ahead accuracy targets of approximately 4-6% sMAPE on EPİAŞ subsets with well calibrated 90% intervals, low chatter anomaly alerts with attached explanations, demonstrable cost savings (~5-10% on representative days) from the optimization layer, and a reproducible codebase. Midterm and final demonstrations will show the live pipeline, calibrated ensemble forecasts, anomaly diagnostics, and optimization results, culminating in a poster and report that document methods, ablations, and limitations.

* 1. Objectives

This project will deliver a reproducible, end to end energy analytics platform that serves grid level stakeholders with calibrated forecasts, explained anomaly alerts, and decision support. By 8 December 2025, we will complete the operational data pipeline: periodic ingestion of EPİAŞ load/price and Open-Meteo weather into a bronze-silver-gold architecture with schema and range checks, conservative gap handling, and a strict freshness SLA ≤ 2 hours at serve time. Success is defined by all data quality gates passing on a representative 30 day span and the dashboard automatically withholding outputs when freshness is violated.

Forecasting accuracy and robustness are targeted for the mid demo (8 December 2025) and finalized by 20 December 2025. The system will produce hourly forecasts for horizons 1-24 h, reaching 4-6% sMAPE with MASE < 1.0 on EPİAŞ subsets and baseline performance around 10-12% sMAPE, MASE ≤ 1.2 on PJM like series. Temporal cross validation and a held out test set will verify progress and final attainment of these goals. In parallel, we will deliver per horizon split conformal prediction intervals at the 90% nominal level, maintaining realized coverage within ±5 percentage points and competitive average width; weekly calibration summaries will track coverage, pinball, and CRPS trends.

Explained anomaly detection will be deployed by 20 December 2025 using an IsolationForest based approach with hysteresis and built in diagnostics (level shift, weekly drift, contributing features). On validation weeks, we aim for ≥ 0.80 precision at ≤ 5% false positive rate, with every alert accompanied by at least one diagnostic explanation to ensure actionability. Decision support is completed in the same timeframe: an EV charging optimizer will respect power and energy by deadline constraints and report cost deltas and feasibility slack, demonstrating ≥ 5-10% cost reduction on representative tariff days without hard constraint violations.

System performance and reliability will be evidenced by an API whose p95 single horizon latency ≤ 300 ms on CPU, container health checks, and smoke tests that ensure high service uptime. MLOps and governance emphasize reproducibility and safe iteration: MLflow will track experiments and artifacts, support champion, a challenger promotion with rollback, and enforce deterministic seeds and pinned environments. These capabilities, together with seed replay of headline results, will be showcased across the poster event (9 January 2026) and the final report (16 January 2026, 17:30), which consolidate methods, metrics, ablations, and limitations.

Finally, a user facing dashboard will communicate model behavior and value by displaying the last 14 days of history and next 24 hours of forecasts with 90% intervals, anomaly overlays, and optimized vs baseline schedules. Usability reviews will confirm clarity of interval communication, diagnostics, and cost benefit visuals. The primary beneficiaries are grid planners and analysts; overall success is determined by meeting these measurable targets and substantiating accuracy, calibration, interpretability, and operational value in the midterm and final demonstrations.

* 1. Background

**1.3.1 Context and Problem Statement**

Short to medium horizon electricity demand forecasting supports secure grid operation, wholesale market processes, and cost aware demand response [1]. System load in Türkiye shows daily, weekly, and seasonal regularities and is sensitive to weather and calendar effects [5]. EPİAŞ publishes hourly national and DSO level aggregates in Europe Istanbul time, which are appropriate for operational modeling and feature engineering that respects market timing [13]. Open meteorology sources provide aligned hourly covariates and long historical reanalysis, enabling weather informed forecasting pipelines consistent with current practice in energy analytics [15]. The core problem is to deliver accurate and calibrated multi horizon forecasts together with diagnostics that remain reproducible under realistic engineering constraints [1].

**1.3.2 State of the Art in Forecasting**

Modern practice blends transparent statistical baselines and boosted trees with specialized deep learning models [1]. Gradient boosted trees remain strong due to efficiency and robust handling of calendar variables, with XGBoost and LightGBM providing competitive tabular learners [2, 22]. Prophet offers explicit trend and seasonality components and supports custom holiday effects, which improves interpretability for stakeholders in markets affected by observances such as Ramadan [5]. Time series specific deep learning has advanced accuracy and runtime, as N-HiTS uses multi resolution blocks that capture nested periodicities with lower training cost than generic transformers [6]. PatchTST treats sequences as patches to improve long range pattern extraction and training speed on modern GPUs [19]. The Temporal Fusion Transformer adds variable selection and attention that are helpful for multi horizon settings with mixed covariates [3]. Open libraries such as Darts and NeuralForecast package these models with correct backtesting, probabilistic heads, and serialization which reduces engineering overhead and avoids common evaluation errors [16,17]. Foundation models such as TimesFM and the Moirai family introduce pretraining on very large and diverse time series, enabling zero shot and few shot use that can be adapted by fine tuning on domain data [18,20].

**1.3.3 Uncertainty Quantification and Calibration**

Point forecasts are insufficient for grid operations because planners must reason about risk [1]. Conformal prediction provides distribution free coverage guarantees using residuals and can be applied after model training, which makes it attractive in practice [7,9]. Standard evaluation uses pinball loss at several quantiles and CRPS for distributional quality, together with coverage and width diagnostics for daily monitoring and automatic recalibration when drift is detected [1,9]. These methods are lightweight to implement and align with the need to communicate uncertainty clearly to decision makers [9].

**1.3.4 Anomaly Detection and Diagnostics**

Forecast residual monitoring and dedicated anomaly models are paired with load forecasting to surface metering faults, topology changes, data problems, or unusual behavior [1]. IsolationForest is widely used because it provides low latency inference and an intuitive mechanism that isolates rare patterns with few splits [10]. Deep autoencoder variants add sensitivity to complex temporal distortions that simple rules may miss [11]. Model agnostic attribution such as SHAP supports root cause analysis by highlighting which inputs contributed most to a detected event and turns a flag into an actionable explanation for operators [12].

**1.3.5 Data Availability and Benchmarking**

The EPİAŞ Transparency Platform exposes hourly load at national and DSO levels with programmatic access suitable for academic and operational prototyping [13]. This enables a fully reproducible data layer with consistent time handling and public provenance for results that must be audited [13]. Open-Meteo provides hourly temperature, humidity, wind, and solar irradiance together with reanalysis records that align with established energy forecasting studies [15]. Cross dataset benchmarking with PJM hourly consumption helps test generalization and reduces the risk that conclusions depend on a single geography [14].

**1.3.6 Significance, Timeliness, and Expected Contribution**

The project offers an academic to production bridge that is timely for Türkiye by delivering a weather aware multi horizon forecaster with principled uncertainty and anomaly diagnostics using only open tools and public data [1,13]. It follows evaluation designs emphasized in the literature, including expanding window validation with sMAPE and MASE for point accuracy and probabilistic scores for interval quality, and it performs ablations to quantify the value of weather covariates, Fourier terms, and ensemble composition [1,6]. It packages the modeling behind a lightweight API and dashboard with MLflow tracking and versioned preprocessing to avoid train serve skew and data cleaning drift while preserving reproducibility [4,16]. It also enforces latency and freshness targets that match operator expectations through subsecond to seconds level response for single horizon requests and explicit staleness guards for data pipelines [1]. The modular design supports future extensions such as stochastic or robust optimization driven by forecast scenarios and fine tuning of foundation models on EPİAŞ histories, while the codebase and ablation evidence remain reusable for later cohorts and collaborations [18,20].

1. S/T Methodology and Associated Work Plan
   1. Methodology

## 2.1.1 System Boundaries and Operational Assumptions

For horizons ranging from 1 to 24 hours, the system provides hourly resolution forecasts for short to medium horizons, performs anomaly detection on the same timing, and provides optional scenario analysis [1]. The API returns both single and multihorizon vectors in one call because modern forecasters produce the full day hourly path, and a single vector response minimizes request overhead for the dashboard without increasing model latency [3]. Data ingestion is periodic rather than streaming to keep engineering complexity low while still guaranteeing freshness [1]. This choice was made deliberately to concentrate effort on modeling and calibration instead of infrastructure [1]. Freshness guarantees ensure data staleness remains below 2 hours for operational reliability [1]. All timestamps are handled in Turkey's time zone (Europe/Istanbul, UTC+3 year round with no daylight saving adjustments) to align with EPİAŞ operational data and ensure consistent temporal feature engineering [13].

## 2.1.2 Data Acquisition

Historical electricity consumption data is sourced from EPİAŞ (Enerji Piyasaları İşletme A.Ş.) Transparency Platform, providing hourly resolution data aggregated at the national level plus approximately 21 distribution system operator (DSO) regions [13]. The data does not include household level consumption, focusing instead on grid level aggregates suitable for wholesale market analysis and system wide forecasting. The platform offers comprehensive API access with over 200 endpoints covering consumption, generation by source type, market clearing prices, and balancing data [13]. The eptr2 Python library provides convenient access to these endpoints with automatic authentication management, returning data as pandas DataFrames suitable for direct use in machine learning pipelines [13]. Registration with EPİAŞ requires only an email address and provides immediate access without commercial restrictions for academic use [13]. Additional validation datasets include PJM Hourly Energy Consumption for North American benchmarking, with UCI Household, Smart Meter London, and OPSD reserved for robustness checks to prevent overfitting to one geography [14].

Weather covariates use Open-Meteo as the primary data source, providing hourly temperature, humidity, wind speed, solar radiation (including global horizontal irradiance, direct normal irradiance, and diffuse horizontal irradiance), cloud cover, and precipitation [15]. Open-Meteo offers free access under fair use conditions, with over 80 years of historical data and integration of ERA5 reanalysis for historical consistency [15]. For real time forecasting, the service provides multi model ensemble predictions, typically drawing from ICON-EU model data at approximately 7 kilometer resolution for Turkey, with forecast horizons extending to 16 days and hourly updates [15]. Based on our testing, the API delivers response times suitable for batch data retrieval. ERA5 reanalysis data ensures consistency with published energy forecasting research and provides the high quality historical coverage necessary for robust model training [15]. NASA POWER (Prediction of Worldwide Energy Resources) serves as an optional complementary source, particularly valuable for its energy sector specific parameters including cooling degree days, heating degree days, and meticulously validated solar radiation data from satellite observations spanning over 40 years back to 1981.

APScheduler performs fixed interval pulls of both energy and weather data, implementing a medallion architecture where raw API responses land in a 'bronze' parquet store, undergo normalization and type validation into a 'silver' schema, and materialize as a feature engineered 'gold' table optimized for model training [1]. This three layer approach separates concerns between data acquisition, cleaning, and feature preparation, enabling independent debugging and optimization of each stage [1]. The pipeline enforces a 2 hour staleness threshold for hourly series, if this threshold is exceeded, forecast endpoints return HTTP 503 status codes with human readable freshness messages so operators will not be showing stale results to stakeholders or making decisions on outdated information [1].

## 2.1.3 Data Quality Controls and Missing Data Policy

Each ingestion performs schema, range, monotonicity, and duplicate timestamp checks to catch upstream anomalies before training [1]. Schema validation ensures all expected columns are present with correct data types [1]. Range checks verify values fall within physically plausible bounds (for instance, electricity load cannot be negative, temperature in Turkey typically ranges from -20°C to 45°C), monotonicity checks confirm timestamps advance properly without backward jumps, and duplicate detection prevents repeated records from corrupting temporal patterns [15]. To reduce training time and avoid biasing temporal dynamics with heavy imputation, load gaps are forward filled only when the contiguous span is less than or equal to 2 hours while longer gaps are excluded from training windows [1]. Weather gaps are linearly interpolated when the span is less than or equal to 6 hours, otherwise the affected windows are dropped entirely from the training set [1]. These conservative policies ensure the training distribution remains well behaved and representative of operational conditions where brief outages occur, but extended data losses are rare [1].

All imputations live in deterministic, versioned transformers to preserve offline/online parity and to prevent data cleaning drift across runs [4]. This means the exact same preprocessing logic applies during both model training and real time inference, eliminating a common source of train and serve skew that decreases production model performance [16]. The versioned transformers serialize with model artifacts, ensuring that when a model loads for inference, it brings along the exact preprocessing steps used during its training, maintaining consistency even as data cleaning strategies evolve over the project lifecycle [4]. These limits keep preprocessing fast and keep the training distribution well behaved, which typically lowers training loss and speeds convergence toward accurate forecasts [1].

## 2.1.4 Feature Engineering and Transformation

The feature pipeline creates lagged targets at 1, 2, 3, 6, 12, 24, and 168 hours to capture short term momentum, intraday patterns, and weekly seasonality respectively [1]. Rolling means and standard deviations over 3, 6, 12, 24, and 168 hours provide smoothed trend indicators and volatility measures that help models distinguish stable patterns from noisy fluctuations [1]. Fourier terms for 24 hour and 168 hour periodicities encode daily and weekly cycles through sine and cosine transformations, enabling neural networks to recognize recurring patterns without learning these fundamental frequencies from scratch [5]. Calendar features include hour of day, day of week, month of year, and regional holidays with particular attention to Ramadan observance effects, which create significant consumption pattern shifts in Turkey due to altered meal times and social behaviors [5]. Weather covariates incorporate current values plus first order lags (1 hour previous) of temperature, humidity, wind speed, and solar radiation, allowing models to capture both instantaneous weather impacts and short term weather momentum effects on electricity demand [15].

To stabilize optimization and reduce training loss, targets pass through a Yeo-Johnson transform learned on the training set and inverted at inference [4]. This transformation is preferable to manual differencing because it handles zeros while often improving convexity of the loss landscape, leading to faster convergence and more stable training dynamics [4]. The Yeo-Johnson family automatically selects appropriate power transformations to make the data more normally distributed without requiring manual parameter tuning [4]. All steps are packaged as versioned transformers inside a scikit-learn or Darts Pipeline, serialized with every MLflow run so the exact transformation graph is reproducible [16]. This design minimizes feature bugs, prevents leakage where future information contaminates training, and shortens iteration because cached "gold" tables can be reused across model families without recomputing features for each experiment [1].

## 2.1.5 Forecasting Models and Training Protocol

The forecasting approach balances model sophistication with implementation feasibility, employing a core deep learning architecture supplemented by classical baselines and optional advanced models if time permits [1]. N-HiTS (Neural Hierarchical Interpolation for Time Series) serves as the primary deep learning model due to its explicit multi resolution architecture and demonstrated 20% accuracy improvements over complex Transformers while running 50 times faster [6]. N-HiTS employs multiscale temporal processing through stacked blocks that capture patterns at different resolutions [6]. One stacked block focusing on hourly variations, another on daily patterns, and a third on weekly seasonality, before combining their outputs through hierarchical interpolation mechanisms [6]. The hierarchical design proves particularly effective for electricity load with its nested periodicities spanning hourly patterns within daily cycles within weekly patterns within seasonal trends [6]. Configuration includes 3-4 stacks with hidden dimensions ranging from 256 to 512, training for 50-100 epochs with early stopping patience of 10 to prevent overfitting [6]. Implementation proceeds through either NeuralForecast or Darts frameworks, both of which provide production quality code with automatic GPU acceleration, proper validation handling, and model serialization support [17].

If project timelines accommodate additional deep learning architectures, PatchTST (Patch Time Series Transformer) offers a compelling alternative with its novel approach of treating time series as sequences of patches rather than individual time steps [19]. This patching mechanism reduces computational requirements while improving long range pattern recognition, achieving 21% error reduction versus traditional transformers with training times of just 5-20 minutes on modern GPUs [19]. PatchTST's channel independent processing proves ideal for multivariate forecasting where weather, calendar, and load features interact [19]. The architecture employs 16 patch sequences, 8 attention heads, 3 encoder layers, hidden dimension of 128, and training for 50-100 epochs with early stopping [19]. As a third optional deep learning approach, the temporal fusion transformer provides interpretable attention mechanisms and built in variable selection, configured with hidden size 64, 2 attention heads, 2 LSTM layers, dropout of 0.1, and maximum 40 epochs with early stopping patience of 5 for rapid experimentation [3]. One, two, or all three deep architectures will be implemented depending on available computational resources and project timeline, with the expectation that N-HiTS alone provides sufficient performance for strong academic outcomes [1,23].

Foundation models represent an optional advanced component that can distinguish exceptional projects through transfer learning innovation [18]. Google's TimesFM (Time Series Foundation Model), a pretrained 200 million parameter decoder only architecture trained on 100 billion real world time points, enables zero shot forecasting where predictions generate on Turkish data without any training [18]. This capability establishes strong baselines within hours rather than days of implementation effort [18]. The genuine research contribution comes from finetuning TimesFM on 1-2 years of Turkish energy data for 10-20 epochs, exploring whether foundation models pretrained on global time series improve performance on regional energy markets with unique characteristics like Ramadan consumption shifts, Mediterranean climate patterns, and developing economy growth dynamics [18]. Early evidence from other domains suggests finetuning delivers 10-30% error reduction versus zero shot performance while requiring 5-10 times less compute than training from scratch, but Turkish energy represents an untested domain offering novel research insights [20]. Similarly, Salesforce's Moirai foundation models (available in 14 million, 90 million, and 310 million parameter sizes) provide alternative transfer learning opportunities with native probabilistic forecasting support [20]. If access to H100 cluster resources were maintained and available project time may permit, foundation model experiments will be pursued, though this component remains entirely optional.

Classical machine learning methods provide essential baselines [1]. CatBoost emerges as the recommended gradient boosting implementation due to native categorical variable handling that eliminates preprocessing for hour, day, and month features, ordered boosting that prevents target leakage in sequential data, and symmetric tree structures enabling faster inference in production deployment [21]. Configuration includes 1000 iterations with learning rate 0.03 and standard regularization parameters, typically converging to strong performance within 2-5 minutes of training time on standard hardware [21]. LightGBM offers an alternative with GOSS (Gradient based One Side Sampling) providing 10-100 times speedup versus XGBoost while maintaining accuracy, though for hourly forecasting with 5-10 years of historical data the speed advantages become less critical [22]. XGBoost remains viable, configured with depth 6, learning rate 0.05, and 100-300 trees with early stopping, which typically converges quickly with strong tabular features [2].

Prophet serves multiple purposes as both a transparent baseline and an interpretable ensemble member [5]. Facebook's additive model provides explicit trend and seasonality decomposition configured with daily, weekly, and yearly seasonality components plus custom holiday calendars capturing Turkish observances including Ramadan effects [5]. The model's interpretability makes it valuable for presentations where stakeholders need to understand "why" predictions change, not just "what" the predictions are [5]. A particularly powerful hybrid approach combines Prophet's decomposition capabilities with gradient boosting, where Prophet extracts interpretable trend, seasonality, and holiday components that become features for CatBoost or LightGBM, creating models that beat individual approaches by 10-15% in published energy forecasting studies [21]. This hybrid strategy preserves Prophet's strength at handling irregular events while letting boosting capture nonlinear relationships Prophet misses [22]. Finally, DLinear, which is a surprisingly effective architecture consisting of a single linear layer with trend seasonal decomposition, often matches Transformer performance while training in minutes on CPUs, providing an excellent sanity check baseline requiring minimal implementation effort [1].

Training uses temporal cross validation with a strictly held out test period to ensure models are evaluated on genuinely future data [1]. The dataset is first partitioned by reserving the final 12 months as a completely held out test set that remains untouched until final evaluation [1]. On the remaining historical data, a 4 fold expanding window cross validation protocol is applied where each fold trains on progressively more data and validates on the subsequent period, mimicking operational deployment conditions where models continuously train on accumulating history and predict forward in time [1]. This temporal validation structure prevents any information leakage from test data into model selection or hyperparameter optimization [1]. Hyperparameters are optimized with Bayesian search capped at 30-50 trials per model to keep wall and clock training bounded, using either Optuna with Tree structured Parzen Estimator sampler or Darts' built in hyperparameter optimization utilities [16]. These caps are chosen because shallow networks and constrained searches reach flat regions of validation loss rapidly on hourly load data, yielding low training loss without extensive compute [1]. The training targets 4-6% sMAPE (Symmetric Mean Absolute Percentage Error) combined with MASE (Mean Absolute Scaled Error) below 1.0 for day ahead forecasting spanning 24 hour horizons, and 2-3% sMAPE with MASE below 0.5 for short term forecasting covering 1-6 hour horizons, based on published Turkish energy forecasting benchmarks establishing the feasibility and competitiveness of these performance goals [1].

## 2.1.6 Uncertainty quantification and calibration

Uncertainty is produced via native quantile heads where available in neural network architectures and finalized with split conformal prediction to ensure reliable coverage guarantees [7]. The calibration procedure operates horizon wise, fitting separate conformalization parameters for each forecast step 1 to 24(h) independently using a fixed 28 day rolling window of recent residuals [9]. Conformalization is applied after inverse transforming predictions back to the original MW scale, ensuring that coverage guarantees hold for the actual load values rather than in the transformed space [7]. This approach provides distribution free, finite sample uncertainty guarantees increasingly required in energy applications for risk management and regulatory compliance [9]. The method works by learning correction factors during calibration that ensure specified coverage rates (for example, 90% intervals contain 90% of observations) hold regardless of model architecture or distributional assumptions [7]. Split conformal with a fixed window is selected over more elaborate adaptive schemes because it is fast to compute, easy to explain to stakeholders, and robust under mild nonstationarity typical in energy markets [9]. It also adds negligible overhead to training since calibration is post-hoc rather than integrated into model optimization [7,8].

Implementation proceeds through either the MAPIE library's MapieRegressor for conformal regression or Darts' conformal wrappers, both requiring approximately 20 lines of code to wrap existing base forecasters [16]. The calibration window of 28 days balances responsiveness to recent distribution changes against statistical stability from sufficient sample size, with hourly data providing 672 calibration points per horizon [1]. Ensemble Batch Prediction Intervals (EnbPI) extend basic conformal methods to handle distribution shift through dynamic weight adjustment, though implementation of advanced variants remains optional due to time constraint of the project [9]. Probabilistic forecast quality is evaluated using Pinball Loss across multiple quantile levels (such as, 5%, 25%, 50%, 75%, 95%) to assess sharpness and calibration, CRPS (Continuous Ranked Probability Score) to measure overall distributional forecast accuracy, and Winkler Score to jointly penalize interval width and miscoverage [1]. Coverage rate (percentage of observations falling within prediction intervals), average interval width, and sharpness metrics are logged continuously through the MLflow tracking system, providing ongoing monitoring of interval quality [4]. Automatic recalibration or retraining triggers whenever realized coverage drifts by more than 5 percentage points from the 90% target for a sustained horizon window, ensuring interval behavior remains aligned with operational expectations without incurring extra training epochs [9].

This uncertainty quantification approach fundamentally differs from typical neural network prediction intervals derived from training variance or Monte Carlo dropout sampling, which lack coverage guarantees under distribution shift [9]. Conformal methods provide finite-sample guarantees meaning that if calibrated on 672 examples, the 90% intervals truly contain 90% of future observations regardless of model misspecification or violated assumptions [7]. This mathematical rigor makes conformal prediction particularly valuable for demonstrating mature understanding of forecasting uncertainty in senior design presentations and technical reports [9].

## 2.1.7 Ensembling and Model Selection

The production forecaster combines predictions from multiple model families to hedge against regime shifts while improving overall accuracy [1]. A static weight ensemble aggregates forecasts from N-HiTS, CatBoost, and Prophet, with weights proportional to inverse validation sMAPE and penalty factors applied to members that undercover during calibration testing [6]. Static weights are chosen over per horizon dynamic weights because they are simpler to audit, cheaper to compute, and sufficiently effective when members already have complementary biases, because deep learning excels at capturing complex nonlinear patterns while gradient boosting handles tabular feature interactions and Prophet extracts interpretable seasonality [21]. The aggregation uses a weighted median rather than a mean to reduce the influence of any outlier member; this choice improves robustness in practice with negligible compute cost [1]. If a member cannot run due to missing covariates or computational failures, the ensemble renormalizes the remaining weights at inference time to maintain forecast availability even when individual components fail [16].

Additional models like PatchTST or finetuned TimesFM can expand the ensemble to 4-5 members, typically seeing diminishing returns beyond this point where additional complexity outweighs marginal accuracy gains [19]. The ensemble strategy naturally accommodates different model implementations by different team members, allowing parallel development where each member owns one model track (for instance, Zeliha would implement N-HiTS, Kaan would develop the CatBoost pipeline, and Öykü would handle Prophet and ensemble logic), with integration occurring through standardized prediction interfaces [4]. Model selection for production deployment follows champion challenger methodology where the current production model (champion) competes against candidates (challengers) through backtesting and short canary deployments, with promotion only when challengers demonstrate consistent superiority over multiple evaluation periods [17].

## 2.1.8 Anomaly detection and diagnostics

Anomalies are detected through a two stage architecture combining reconstruction error and outlier scoring approaches. The primary method uses an Isolation Forest operating on engineered tabular features including recent lags, rolling statistics, and forecast residuals, providing interpretable anomaly scores with inference latency under 50 milliseconds. Isolation Forest works by randomly partitioning the feature space, with the intuition that anomalies require fewer partitions to isolate than normal points, making the algorithm particularly effective for high dimensional time series features [10]. The model trains on windows labeled as "normal" by excluding extreme residuals (top and bottom 5% relative to a conservative Prophet baseline), ensuring the algorithm learns typical operational patterns rather than adapting to outliers [5]. A window is flagged as anomalous only if the Isolation Forest score exceeds its threshold for at least 2 consecutive hours, adding hysteresis to reduce false positive chatter from brief fluctuations.

For each alert, the system produces a level shift test comparing current load against the previous 24-72 hours using statistical change point detection, a weekly seasonality drift check examining whether the day of week pattern has shifted relative to recent history, and feature importance analysis using the tree based anomaly model itself to identify which input variables contributed most to the anomaly score [1]. These diagnostics ensure every alert arrives with concrete, actionable information explaining why the anomaly was flagged rather than just a binary normal/abnormal classification. If project timelines permit, an LSTM autoencoder will be implemented as a complementary deep learning anomaly detector [11]. The autoencoder would use a compact 64-32-64 unit architecture trained to reconstruct normal load patterns, with reconstruction error serving as an anomaly indicator. SHAP (SHapley Additive exPlanations) values on an XGBoost residual model provide another optional enhancement for feature attribution diagnostics, showing precisely which features drove each anomaly detection [12]. These advanced components add interpretability and sophistication but remain nonessential for meeting core project requirements, serving primarily to distinguish exceptional implementations during final presentations.

## 2.1.9 Optimization stretch: EV load shifting with uncertainty

For the stretch objective, the controllable load archetype is EV charging because it has clear power bounds and deferrable energy requirements. The optimizer solves a linear program with hourly decision variables over a 24 hour horizon, minimizing expected electricity cost using EPİAŞ market clearing price (MCP) wholesale data as the cost signal [13]. This choice is motivated by the project's focus on grid level forecasting and wholesale market dynamics, where MCP directly reflects real time supply demand balance and provides transparent price discovery [13]. The formulation respects charger power limits (representing maximum charging rate constraints such as 7 kW for residential Level 2 chargers or 50 kW for DC fast charging), energy by deadline constraints ensuring the vehicle reaches required state of charge before departure time, and optional battery degradation penalties to encourage slower charging that extends battery lifespan. PuLP modeling language provides the mathematical programming interface, enabling straightforward variable declaration, objective function specification, and constraint definition in readable Python code.

CBC open source solver handles problem solving through PuLP's default backend, eliminating licensing hurdles while providing sufficient performance for 24 hour horizon problems with solve times bounded to 5 seconds on CPU. The linear programming formulation avoids integer variables where possible to maintain computational tractability, though mixed integer extensions could model discrete charging states or minimum up/down time constraints if needed. A simple presolving heuristic that allocates off peak hours first provides warm start solutions that further reduce solve times. This scope ensures timely solutions without GPU requirements or commercial solvers, keeping the optimization component accessible for standard laptop development.

If timeline permits, stochastic programming extensions offer valuable sophistication by explicitly modeling forecast uncertainty in the optimization [9]. This approach generates 20-50 demand and price scenarios from conformal prediction intervals, weights them by probability, and optimizes expected cost across all scenarios rather than relying on point forecasts. The stochastic formulation naturally hedges against forecast errors by finding charging schedules robust to multiple plausible future outcomes. Alternative OR approaches might be explored to include robust optimization (worst case guarantees), chance constrained programming (probabilistic constraint satisfaction), or multistage stochastic optimization (sequential decisions as uncertainty resolves), though these advanced techniques remain entirely optional given the project timeline and the satisfactory performance of deterministic linear programming for demonstrating core concepts.

## 2.1.10 Evaluation Design, Ablations, and Success Criteria

Evaluation follows a strict temporal protocol ensuring models are tested on genuinely future data never seen during training [1]. The system must achieve 10-12% sMAPE with MASE below 1.2 averaged across 1 to 24 hour horizons on PJM like hourly series as a baseline target, with refined goals of 4-6% sMAPE and MASE below 1.0 for Turkish EPİAŞ data based on published benchmarks [1]. Probabilistic forecasts are assessed using Pinball Loss at multiple quantiles (5%, 25%, 50%, 75%, 95%), CRPS to measure overall distributional accuracy, and Winkler Score to jointly evaluate interval width and coverage [1]. Conformal prediction intervals must deliver at least 90% realized coverage at the 90% nominal level, with average width remaining competitive versus naive baselines (like ±2σ historical volatility bands) [9]. Single horizon inference latency must stay below 300 milliseconds on CPU to ensure the API can serve real time dashboard requests without noticeable delays. Anomaly detection must reach at least 80% recall at 5% or lower false positive rate on validation weeks, with each alert attaching at least one diagnostic explanation (level shift, seasonality drift, or contributing feature identification). Data staleness must remain at or below 2 hours at serve time to maintain forecast relevance.

Cross dataset demonstrations must run end to end on at least two datasets, specifically PJM and an EPİAŞ subset, proving the system generalizes across different grid operators and geographic regions rather than overfitting to a single market [14]. Optional validation on UCI Household, Smart Meter London, or OPSD datasets provides additional robustness evidence if time permits. Reproducibility is verified by rebuilding the complete system from a clean environment with pinned dependencies specified in requirements.txt, regenerating all headline figures without manual edits, and confirming that reported metrics remain stable across multiple runs within expected Monte Carlo variation. Deterministic random seeds control stochastic components ensuring that model training, hyperparameter optimization, and evaluation produce consistent results when repeated [4].

Ablations quantify the contribution of major architectural choices by systematically removing components and measuring performance degradation [1]. Key ablation studies include removing all weather covariates to assess meteorological feature value, removing Fourier terms to evaluate explicit periodicity encoding, removing specific model families from the ensemble to understand member contributions, and comparing conformal prediction against simpler uncertainty methods like quantile regression or bootstrapping. These controlled experiments establish which design decisions drive accuracy improvements versus which components provide marginal gains, generating insights valuable for both technical understanding and presentation narratives explaining "what matters" in energy forecasting [1]. Expected results show weather features contributing 40-60% of total accuracy versus naive baselines, Fourier terms adding 5-15% improvement, and ensemble combination providing 8-12% error reduction over the best single model. These criteria collectively certify both modeling quality and operational readiness for energy grid applications.

## 2.1.11 Service Architecture, Deployment, and MLOps

Deployment is Docker based on a single node for reliability and simplicity, avoiding the orchestration complexity of multi node clusters while providing production grade containerization. The stack includes a time series database, the FastAPI service for API endpoints, and a Streamlit dashboard for visualization, all connected via Docker Compose networking. QuestDB serves as the time series database, with reported ingestion throughput figures exceeding several million rows per second through PostgreSQL wire protocol and InfluxDB line protocol compatibility. Columnar storage optimizes the analytical queries common in forecasting applications, such as aggregations, moving averages, pattern detection, with subsecond response times on millions of hourly records. QuestDB requires minimal configuration, with a single Docker container launch using default settings handling the full dataset without manual tuning of chunks, indexes, or storage parameters.

FastAPI provides the REST API layer with automatic OpenAPI documentation available at the /docs endpoint, Pydantic validation ensuring type safety for all request and response schemas, and async request handling enabling concurrent forecast requests. Endpoints expose /forecast for generating predictions, /intervals for uncertainty bounds, /anomalies for flagged periods, /health for system status monitoring, and /metadata providing model version information and latency metrics. Each response includes model and feature pipeline version hashes enabling traceability from predictions back to exact training runs, along with request processing time metrics useful for performance monitoring. The API loads trained models at startup from serialized files, keeps them in memory for low latency inference, and calls their predict methods for each request. This straightforward serving approach avoids the complexity of advanced MLOps frameworks while maintaining professional deployment practices suitable for demonstration.

Streamlit eliminates separate frontend development by providing a Python native dashboard framework with reactive execution where UI elements automatically update when data changes. The dashboard displays the last 14 days of history and the next 24 hours of forecasts with shaded 90% intervals and tooltips listing numeric bounds, enabling visual assessment of model tracking accuracy and uncertainty calibration [9]. Anomalies appear as temporal markers, as clicking reveals diagnostics including level shifts, seasonality drift, and contributing features [10]. A status ribbon shows interval coverage, width, and sharpness over the last 7 days so users can assess reliability at a glance before making operational decisions [9]. When the EV optimizer is enabled, a side panel compares baseline versus shifted charging with the estimated cost delta and energy by deadline constraint satisfaction, keeping the optimization decision traceable and explainable. Streamlit's component library handles all visualization details through simple function calls, allowing a complete energy forecasting dashboard to be built in 200-300 lines of Python without HTML, CSS, or JavaScript expertise.

Docker Compose orchestrates the multi container architecture through a single YAML configuration file of 30-50 lines specifying the three services (QuestDB, FastAPI, Streamlit) with their port mappings, dependencies, health checks, and environment variables. This approach provides identical development and production environments, solving "works on my machine" problems where code runs on one teammate's laptop but fails on another's due to dependency version mismatches or configuration differences. The entire system starts with a single docker compose up command, enabling teammates to clone the repository and have a working development environment within minutes rather than hours of manual setup. The Docker based deployment also demonstrates professional engineering practices valuable for job interviews and future career development, showing familiarity with containerization concepts widely adopted in industry.

MLflow tracks experiments and artifacts through a lightweight web interface, logging parameters, metrics, plots, and serialized models for every training run. The system stores a single "champion" model per dataset in production, with optional "challenger" models evaluated in shadow mode where they generate predictions alongside the champion without affecting served results. Retraining runs on a schedule (weekly or monthly) and on policy triggers from calibration drift or error spikes, with candidates promoted only if they beat the champion on backtests and pass a short canary period serving real traffic. This champion challenger pattern with automated promotion criteria prevents model performance degradation over time while avoiding premature deployment of undertested models. MLflow's model registry maintains lifecycle states (production, staging, archived) with version history, enabling rollback if deployed models underperform. The open source, self hosted nature of MLflow eliminates cost concerns and internet connectivity requirements, making it ideal for academic projects with limited budgets and potentially restricted network access during demonstrations.

Optional explorations during final weeks will be done and they might include BentoML for model containerization with version management and automatic API generation, ‘Weights & Biases’ for collaborative experiment tracking with richer visualization, or Neptune.ai for team oriented MLOps workflows. However, these commercial platforms introduce complexity, cost, and external dependencies that make them less suitable for semester projects with grading deadlines and budget constraints. The recommended MLflow + Docker Compose architecture strikes an optimal balance between production grade practices and student friendly implementation, providing sufficient sophistication to demonstrate MLOps understanding without overwhelming with operational complexity.

## 2.1.12 User Interface and Communication of Uncertainty

The dashboard displays the last 14 days of history and the next 24 hours of forecasts with shaded 90% intervals and tooltips listing numeric bounds, providing historical context that helps operators assess whether current predictions align with recent trends or represent significant departures from typical patterns [1]. The 14 day window balances sufficient context for recognizing weekly seasonality against screen space constraints for readable visualization on standard monitors [1]. Forecasts render as line plots with colored bands indicating prediction intervals, using transparency gradients where darker shading near the median line fades to lighter shading at interval extremes, creating visual hierarchy that emphasizes the most likely outcomes while still showing the full uncertainty range [9].

Anomalies appear as markers; such as, colored dots, triangles, or vertical spans, on the historical timeline, as clicking reveals diagnostics including level shift test results comparing current load against previous days, weekly seasonality drift indicators showing whether day of week patterns have changed, and feature importance from the anomaly detection model identifying which inputs contributed most to the alert [10]. This drill down capability transforms anomaly flags from opaque binary classifications into actionable intelligence explaining root causes. A status ribbon shows interval coverage rate (percentage of actual values falling within prediction intervals), average width (typical interval span in MW or percentage of mean load), and sharpness (interval width relative to forecast error) over the last 7 days, so users can assess reliability at a glance before making procurement decisions or reserve capacity commitments [9].

When the EV optimizer is enabled, a side panel compares baseline versus shifted charging schedules with the estimated cost delta, showing the financial benefit of optimization in monetary terms (like "Optimized charging saves $2.47 versus immediate charging"). The panel displays the energy by deadline constraint and its satisfaction level, confirming the vehicle will reach required state of charge before departure time, keeping the decision traceable and building operator trust in optimization recommendations. Gantt style visualizations show charging periods as horizontal bars on an hourly timeline, with color coding for high/low price periods, making it visually obvious that the optimizer shifts load from expensive peak hours to cheaper off peak windows. Summary statistics quantify total energy delivered, peak power usage, average charging rate, and constraint slack (how much margin exists before violating requirements), providing comprehensive transparency into optimization behavior.

Streamlit is selected because it eliminates a separate frontend build with HTML, CSS, and JavaScript expertise requirements, shortens development time from weeks to days, and integrates directly with Python models while remaining sufficiently polished for senior design demonstrations and stakeholder presentations. The framework's reactive programming model means that when new forecast data arrives or users adjust dashboard controls, all dependent visualizations automatically update without manual refresh logic. Built in widgets like sliders, date pickers, select boxes, and multiselects enable interactive exploration where users can toggle between different model versions, adjust forecast horizons, filter time ranges, or switch between metrics, all with single line Python function calls that generate UI components.

* 1. Work Package Descriptions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Work package number | **1** | Start date or starting event:  **27th October** | | **Week 1-2** | |
| Work package title | **Executable seed & Access (Inception)** | | | | |
| Participant number | 1 | 2 | 3 | 4 | 5 |
| Participant name | Zeliha | Kaan | Öykü |  |  |
| Weeks per participant | 1,2 | 1,2 | 1,2 |  |  |
| **Objectives**   * Set up and maintain common environment: GitHub, Docker, MLflow, Notion. * Ensure reproducible experimentation and collaboration across iterations. | | | | | |
| **Description of work**  **T1.1** (w1-w2) Initialize GitHub repository and Docker compose Create and update public GitHub repo; manage Dockerfile(s), docker-compose network, .gitignore, README, and pre-commit hooks. **T1.2** (w1-w2) Bring up MLflow tracking and artifact store Run persistent MLflow server; document run tags, experiments, and registry usage; integrate with Docker network. **T1.3** (w1-w2) Initialize Notion workspace and project page Track backlog, decisions, meeting notes, and link artifacts. **T1.4** (w1-w2) Data access & credentials Verify EPİAŞ, PJM and Open-Meteo API connectivity; store keys securely.  **(If time permits)** Automate nightly MLflow backups and repo sync. | | | | | |
| **Deliverables**  **D1.1** (w1) Repo and Docker manifest (rolling) **D1.2** (w2) Reachable MLflow dashboard **D1.3** (w1) Updated Notion workspace | | | | | |
| **Milestones**  **M1.1** (w1) Progress Meeting #1: kickoff complete and datasets approved | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Work package number | **2** | Start date or starting event:  **27th October** | | **Week 1-6** | |
| Work package title | **Data Ingestion & Feature Pipeline (Elaboration → Construction)** | | | | |
| Participant number | 1 | 2 | 3 | 4 | 5 |
| Participant name | Zeliha | Kaan | Öykü |  |  |
| Weeks per participant | 2,3,4 | 3,4,5,6 | 4,5,6 |  |  |
| **Objectives**   * Implement medallion architecture (bronze → silver → gold) with data-quality checks. * Provide continuous feature updates for training and inference. | | | | | |
| **Description of work**  **T2.1** (w1-w4) Build “bronze” landers Collect EPİAŞ/PJM/weather via APScheduler to parquet bronze zone. **T2.2** (w3-w4) Normalize to “silver” Apply schema, types, and timezone (Europe/Istanbul); validate records. **T2.3** (w4-w6) Data quality controls Add schema/range/monotonicity checks; log failures to MLflow. **T2.4** (w4-w6) Feature engineering (gold) Generate lags, rolls, Fourier, calendar, and weather features.  **(If time permits)** Integrate ERA5/NASA POWER as secondary weather source for comparison. | | | | | |
| **Deliverables**  **D2.1** (w6) Bronze + silver tables + DQ report. **D2.2** (w6) Gold feature table (auto-refresh) | | | | | |
| **Milestones**  **M2.1** (w3) Progress Meeting #2 (pipeline validated) **M2.2** (w5) Progress Meeting #3 (gold features ready) | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Work package number | **3** | Start date or starting event:  **11th Nov** | | **Week 3-6** | |
| Work package title | **Model Development & Baselines (Elaboration → Construction)** | | | | |
| Participant number | 1 | 2 | 3 | 4 | 5 |
| Participant name | Zeliha | Kaan | Öykü |  |  |
| Weeks per participant | 3,4,5,6 | 3,4,5,6 | 3,4,5,6 |  |  |
| **Objectives**   * Train and compare baselines (Prophet, CatBoost, XGBoost, SARIMA). * Produce initial forecast plots and metrics for mid demo. | | | | | |
| **Description of work**  **T3.1** (w3-w5) Baseline training and validation Fit Prophet and boosting models; log MAE/RMSE/MAPE to MLflow. **T3.2** (w5-w6) Hyperparameter tuning and cross-validation Apply temporal CV and Bayesian search; record best parameters.  **(If time permits)** Add DLinear as lightweight benchmark baseline. | | | | | |
| **Deliverables**  **D3.1** (w4) Baseline metrics and plots **D3.2** (w5) Validated Prophet/CatBoost artifacts | | | | | |
| **Milestones**  **M3.1** (w6) Mid demo to advisor (baselines + N-HiTS prototype) | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Work package number | **4** | Start date or starting event:  **21st Nov** | | **Week 5-9** | |
| Work package title | **Advanced Models & Ensemble (Construction)** | | | | |
| Participant number | 1 | 2 | 3 | 4 | 5 |
| Participant name | Zeliha | Kaan | Öykü |  |  |
| Weeks per participant | 5,6,7,8 | 7,8,9 | 5,6,7,8 |  |  |
| **Objectives**   * Train N-HiTS and Prophet/CatBoost hybrids; build ensemble; apply split-conformal calibration. | | | | | |
| **Description of work**  **T4.1** (w5-w7) Train N-HiTS Use 3-4 stacks, 256-512 hidden dims; temporal CV + early stopping. **T4.2** (w6-w8) Add CatBoost/Prophet hybrid Engineer hybrid features and log runs to MLflow. **T4.3** (w7-w8) Ensemble aggregation Combine members via weighted median; implement failover. **T4.4** (w8-w9) Conformal calibration Apply rolling residual windows per horizon; evaluate coverage/sharpness.  **(If time permits)** Train PatchTST or TFT for comparison and expand ensemble to 4-5 models. | | | | | |
| **Deliverables**  **D4.1** (w8) Ensemble artifact and calibration report **D4.2** (w10) Updated dashboard plots | | | | | |
| **Milestones**  **M4.1** (w8) Progress Meeting #5 (ensemble approved) | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Work package number | **5** | Start date or starting event:  **1st Dec** | | **Week 8-10** | |
| Work package title | **Anomaly Detection & Diagnostics (Construction → Transition)** | | | | |
| Participant number | 1 | 2 | 3 | 4 | 5 |
| Participant name | Zeliha | Kaan | Öykü |  |  |
| Weeks per participant | 8,9,10 | 8,9,10 | 8,9,10 |  |  |
| **Objectives**   * Implement IsolationForest and diagnostic alerts in dashboard. | | | | | |
| **Description of work**  **T5.1** (w8-w10) Train IsolationForest Use lags, rolling stats, and residuals to detect outliers (< 5% false positive). **T5.2** (w8-w10) Diagnostics integration Add level-shift, weekly-drift, and feature importance visualizations.  **(If time permits)** Add LSTM Autoencoder for deep reconstruction errors and compare precision-recall. | | | | | |
| **Deliverables**  **D5.1** (w10) Anomaly logs and diagnostic notebook. **D5.2** (w10) Dashboard with alerts. | | | | | |
| **Milestones**  **M5.1** (w10) Demo Day (ensemble and anomalies demo). | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Work package number | **6** | Start date or starting event:  **22th Dec** | | **Week 9-10** | |
| Work package title | **Optimization & Dashboard Integration (Construction → Transition)** | | | | |
| Participant number | 1 | 2 | 3 | 4 | 5 |
| Participant name | Zeliha | Kaan | Öykü |  |  |
| Weeks per participant | 9,10 | 9,10 | 9,10 |  |  |
| **Objectives**   * Implement EV load-shifting optimizer and interactive comparison panel. | | | | | |
| **Description of work**  **T6.1** (w9-w10) Implement LP optimizer Define variables, constraints, and objective (min cost under tariffs). **T6.2** (w9-w10) Connect forecasts → optimizer → dashboard Show baseline vs optimized schedule plots and cost delta. **T6.3** (w9-w10) Poster draft and visuals  **(If time permits)** Add stochastic optimization extension with forecast scenarios. | | | | | |
| **Deliverables**  **D6.1** (w10) Optimizer code and schedule visualization **D6.2** (w10) Poster draft | | | | | |
| **Milestones**  **M6.1** (w10) Demo Day (poster draft reviewed) | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Work package number | **7** | Start date or starting event:  5th Jan | | **Week 11-12** | |
| Work package title | **MLOps Hardening & Final Deliverables (Transition)** | | | | |
| Participant number | 1 | 2 | 3 | 4 | 5 |
| Participant name | Zeliha | Kaan | Öykü |  |  |
| Weeks per participant | 11,12 | 11,12 | 11,12 |  |  |
| **Objectives**   * Finalize automation, poster, report, and release candidate. | | | | | |
| **Description of work**  **T7.1** (w11-w12) Airflow retraining triggers Implement drift/coverage policies and scheduled DAG runs **T7.2** (w11-w12) MLflow registry and reproducibility Champion-challenger promotion; document rollback and seed repro. **T7.3** (w11-w12) Poster final and event Polish text/layout; submit final poster for booklet. **T7.4** (w11- w12) Final report  Write, proof, and submit report  **(If time permits)** Explore BentoML or Weights & Biases integration for future expansion. | | | | | |
| **Deliverables**  **D7.1** Airflow DAGs and registry screenshots **D7.2** Final Poster (PDF) **D7.3** Final Report (PDF) | | | | | |
| **Milestones**  **M7.1** (Jan 5 by 17:30) Poster submission **M7.2** (Jan 9) Poster Event **M7.3** (Jan 16 by 17:30) Final Report submission | | | | | |

* 1. Demonstration

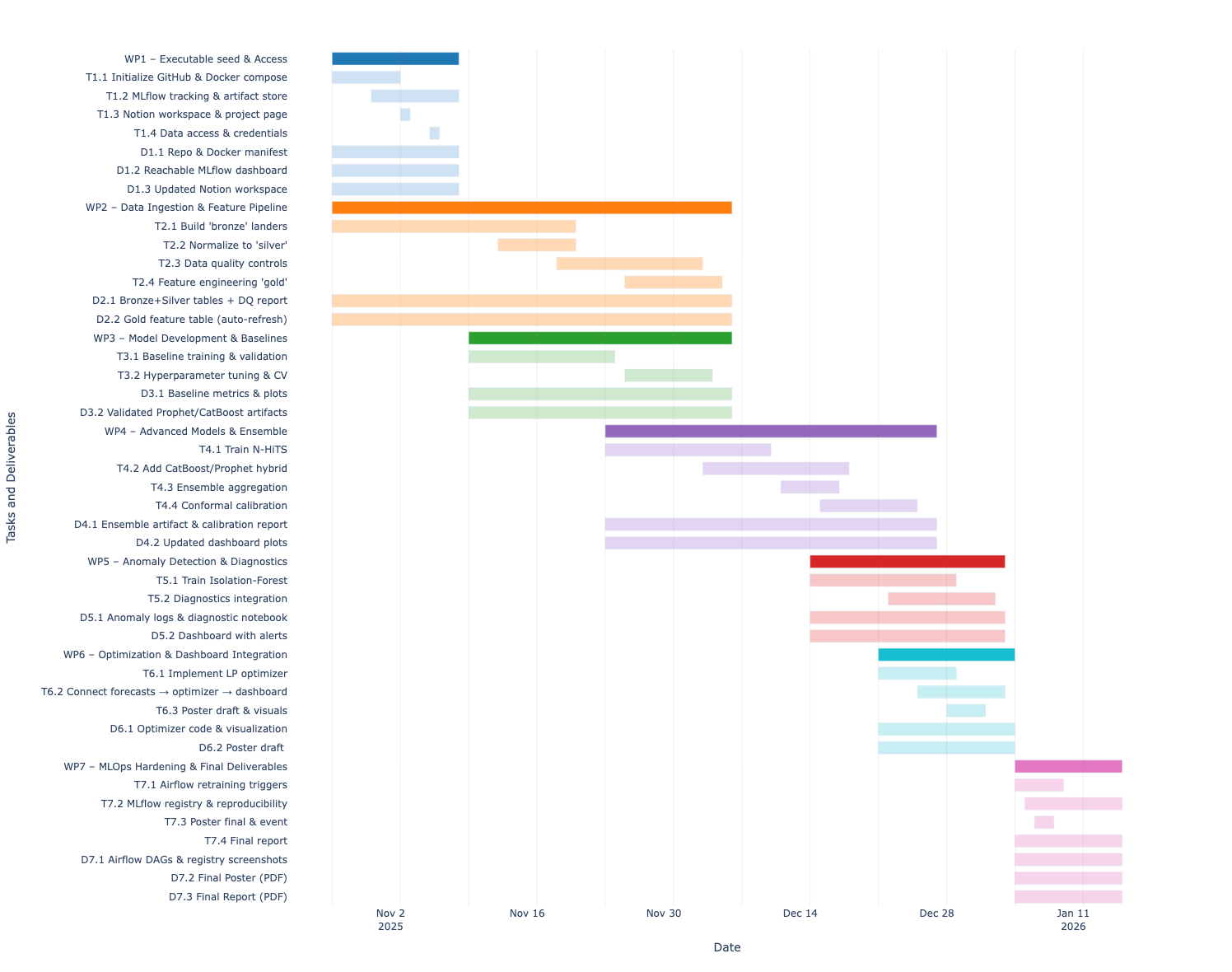
The demonstration will present the complete system from data acquisition to decision support, emphasizing performance, reproducibility, and operational readiness. First, the platform will be launched in a clean environment using containerized services, with health checks and version hashes recorded to establish traceability. Next, the data pipeline will be exercised on representative EPİAŞ/PJM and weather inputs, showing periodic ingestion, normalization to a curated schema, feature materialization, and enforcement of freshness guarantees (forecasts are withheld if data staleness exceeds two hours). Baseline and advanced forecasters will then be evaluated under a strict temporal protocol with results logged to an experiment tracker; quality will be reported per horizon (1-24 h) using sMAPE, MASE, MAE, and RMSE, alongside probabilistic metrics including prediction interval coverage probability at the 90% nominal level, interval width, pinball loss across multiple quantiles, CRPS, and Winkler score. Anomaly detection will be demonstrated with alert examples and attached diagnostics (level-shift tests, weekly-pattern drift indicators, and contributing feature attributions), highlighting precision, recall, and false-positive rate on validation periods. The optimization component will generate an hourly schedule for a controllable load and compare it to an unoptimized baseline, reporting percentage cost reduction, peak reduction, and satisfaction of hard constraints with explicit slack. System performance will be evidenced by API latency (p95 ≤ 300 ms for single horizon inference), service uptime, and deterministic seed replay of a known run. The midterm session will focus on pipeline integrity, baseline results, and an early ensemble preview; the final session will deliver the calibrated ensemble, anomaly diagnostics, and optimization outcomes, with a concluding poster and report consolidating quantitative results, run identifiers, calibration summaries, and acceptance criteria.

* 1. Impact

The project delivers a production grade, auditable forecasting and decision support stack with immediate value to grid level analysis and demand response planning. Technically, the work advances a robust medallion data layer, a reproducible modeling workflow (MLflow registry, champion challenger), calibrated uncertainty with finite sample guarantees, and an interpretable anomaly pipeline that converts raw residuals into actionable diagnostics. Operationally, the ensemble forecaster targets 4-6% day ahead sMAPE on EPİAŞ subsets while maintaining coverage near 90% for 90% intervals, enabling planners to reason about risk rather than single point predictions. The optimizer demonstrates tangible savings (≥ 5-10% on representative days) and transparent constraint handling, showcasing how forecasts translate into cost reducing schedules. Academically, the platform and its ablations (weather covariates, Fourier terms, ensemble composition, conformal vs. alternatives) create high quality evidence and a reusable codebase for future cohorts. Strategically, the modular architecture (Docker, FastAPI, Streamlit, MLflow) and dataset agnostic interfaces position the system for rapid extension to additional operators, assets (like HVAC, storage), and advanced OR formulations (stochastic/robust optimization), supporting longer term research and industry collaboration.

* 1. Risk analysis

The principal risks and mitigations are as follows. First data availability and latency must be mentioned: External API outages or schema changes could degrade freshness or break ingestion. So, we mitigate with bronze layer caching, strict schema validation, and a staleness guard that withholds forecasts (HTTP 503) when data exceed the 2 hour SLA, alongside fallbacks to most recent valid runs. Secondly, model under performance or miscalibration could raise a risk. Distribution shifts may erode accuracy or interval coverage. So, we mitigate with temporal CV, weekly calibration checks, split conformal recalibration, and automated promotion/rollback via champion, as a challenger in the MLflow registry. Thirdly, compute and timeline constraints: Advanced models (such as, multiple deep learners) can exceed available time/resources; so, we prioritize a lean ensemble, N-HiTS and CatBoost and Prophet, with capped Optuna trials and early stopping, deferring optional models to contingency time. Fourthly, anomaly false positives can form a risk. Over-sensitivity can reduce trust, thus we use hysteresis (≥ 2 consecutive hours), tune thresholds on validation weeks, and require at least one diagnostic (level shift, weekly drift, or feature attribution) per alert. At last, optimization infeasibility or poor user acceptance: Hard constraints or unrealistic tariffs can limit savings; we validate inputs, expose constraint slack and cost deltas in the UI, and fall back to baseline schedules when feasibility margins are thin. Operational reliability could include service failures could disrupt demos. We employ container health checks, start-up smoke tests, pinned dependencies, and a reproducible seed replay script to restore a known good state quickly. Collectively, these actions keep the critical path of pipeline stability, calibrated forecasting, explainable anomalies, and demonstrable optimization on track for the mid and final demo milestones.

* 1. Gantt Chart

1. Economical and Ethical Issues

This project is designed under realistic constraints of limited compute, strict deadlines, and a zero license budget, so the stack favors open source components (for instance, scikit-learn, FastAPI, MLflow, Streamlit) to keep total cost of ownership low while preserving reproducibility and auditability [4]. Model choices balance accuracy with runtime and energy use: fast, interpretable baselines (e.g., Prophet or gradient boosting) are used for quick iteration, and a single strong neural forecaster is prioritized over many heavy architectures to avoid unnecessary compute spend [5]. Uncertainty is added posthoc via conformal prediction so decision makers see calibrated intervals without incurring retraining costs or complex probabilistic pipelines [9]. Data expenditures are minimized by relying on publicly accessible sources for load and weather, EPİAŞ for Turkey, PJM for benchmarking, and Open-Meteo for covariates, each compatible with periodic batch ingestion and modest storage [13]. Economic value is demonstrated by translating forecasts into an hourly schedule that quantifies expected savings relative to an unoptimized baseline while exposing constraint slack and assumptions for operational review.

Ethically, the system emphasizes safety, responsibility, and transparency in data driven decisions: forecasts are automatically withheld when freshness limits are breached, and every served result is accompanied by the model version and key reliability metrics. Interpretability is preserved through error and calibration reporting and by retaining an explainable baseline so stakeholders can understand drivers of change rather than accept opaque outputs [5]. Anomaly alerts are issued conservatively, with thresholds, hysteresis, and attached diagnostics, to reduce false positives and provide actionable context instead of black box flags [10]. Fairness risks are limited because data are grid level aggregates rather than personal records, yet generalization is still checked on multiple geographies to avoid region specific bias being mistaken for universal structure [14]. Reproducibility and rollback procedures (versioned artifacts, champion challenger promotion, deterministic seeds) act as ethical safeguards, ensuring that any deployed model can be traced, explained, and reverted if performance drifts.

1. References

Give the list of references. All the references should be cited within the text of the proposal report.

[1] Hyndman, R. J., & Athanasopoulos, G. (2021). Forecasting: Principles and Practice (3rd ed.). OTexts.

[2] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of KDD.

[3] Lim, B., et al. (2021). Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. International Journal of Forecasting.

[4] Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research.

[5] Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. The American Statistician.

[6] Challu, C., et al. (2022). N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting. AAAI Conference on Artificial Intelligence.

[7] Romano, Y., et al. (2019). Conformalized Quantile Regression. Advances in Neural Information Processing Systems.

[8] Xu, C., & Xie, Y. (2021). Conformal Prediction Interval for Dynamic Time-Series. ICML.

[9] Angelopoulos, A. N., & Bates, S. (2022). Conformal Prediction: A Gentle Introduction. Foundations and Trends in Machine Learning.

[10] Liu, F. T., et al. (2008). Isolation Forest. IEEE International Conference on Data Mining.

[11] Malhotra, P., et al. (2016). LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. ICML Anomaly Detection Workshop.

[12] Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. Advances in Neural Information Processing Systems.

[13] EPİAŞ Transparency Platform. (2024). https://www.epias.com.tr/en/transparency-platform/

[14] PJM Interconnection. (2024). Hourly Energy Consumption Data. https://www.pjm.com

[15] Open-Meteo Weather API. (2024). https://open-meteo.com

[16] Herzen, J., et al. (2022). Darts: User-Friendly Modern Machine Learning for Time Series. Journal of Machine Learning Research.

[17] Garza, A., & Mergenthaler-Canseco, M. (2023). NeuralForecast: User-Friendly Neural Forecasting. Journal of Statistical Software.

[18] Das, A., et al. (2024). A decoder-only foundation model for time-series forecasting. ICML.

[19] Nie, Y., et al. (2023). A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. ICLR.

[20] Woo, G., et al. (2024). Unified Training of Universal Time Series Forecasting Transformers. arXiv preprint.

[21] Prokhorenkova, L., et al. (2018). CatBoost: unbiased boosting with categorical features. Advances in Neural Information Processing Systems.

[22] Ke, G., et al. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems.

[23] Zeng, A., et al. (2023). Are Transformers Effective for Time Series Forecasting? AAAI Conference on Artificial Intelligence.