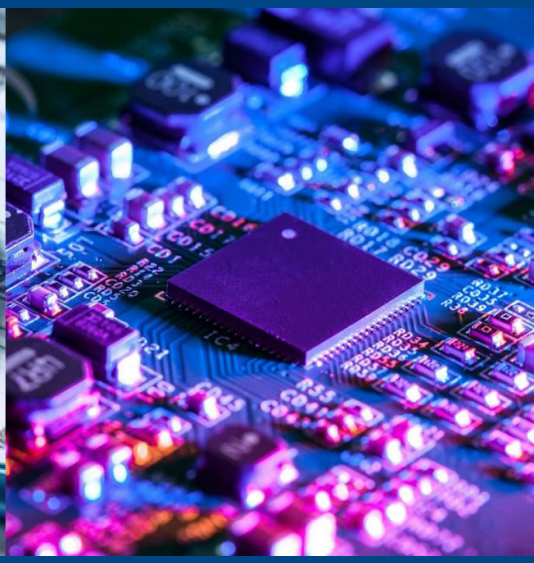
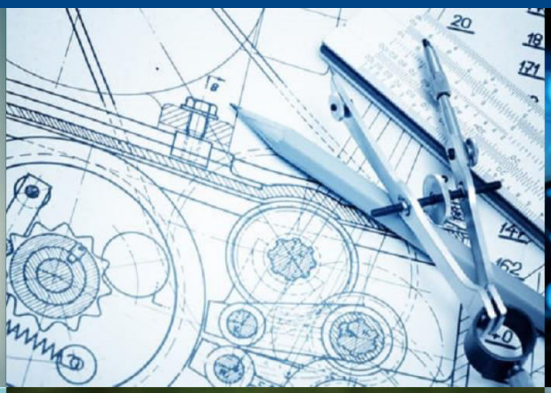
ISSN: 2582-7219

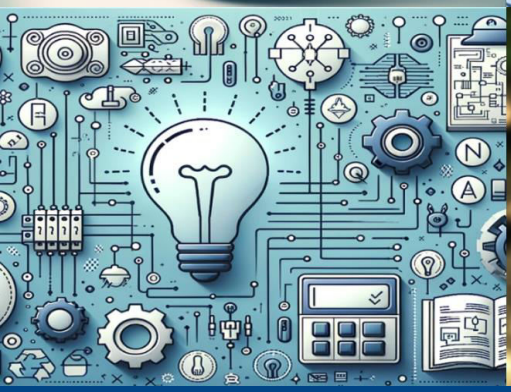




International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.206 **Volume 9, Issue 4, April 2026**

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904176

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Smart Parking Spot Prediction System using AWS Cloud, Time-Series Forecasting, and Generative AI

Abdulkalam J[1], Abieshwar T[1], Bharani Dharan T[1], Mr. G. Dhanapathy[2]

Department of Artificial Intelligence and Data Science, Sri Manakula Vinayagar Engineering College, Puducherry, India[1]

Assistant Professor, Department of Artificial Intelligence and Data Science, Sri Manakula Vinayagar Engineering College, Puducherry, India[2]

ABSTRACT: Urban parking congestion is a persistent challenge in modern cities, leading to increased fuel consumption, traffic delays, environmental pollution, and driver frustration. Conventional parking systems rely on static infrastructure with no predictive capability, making them ill-suited for dynamic demand patterns. This paper presents the design and implementation of a Smart Parking Spot Prediction System that integrates AWS cloud infrastructure, time-series machine learning, and Generative AI to deliver intelligent, real-time parking management. The proposed system ingests occupancy data from IoT sensors through AWS API Gateway and Lambda functions, persists time-series readings in Amazon Timestream, archives raw data in Amazon S3, and performs historical querying via Amazon Athena. The Facebook Prophet model is deployed within AWS Lambda to forecast hourly parking occupancy, enabling a rule-based dynamic pricing engine that adjusts rates across four demand tiers. An Amazon Bedrock-powered Generative AI assistant provides natural language navigation guidance and availability recommendations to drivers and pricing insights to operators. The system is designed to significantly improve parking space utilization, reduce circling-related congestion, and maximize operator revenue through adaptive pricing. This architecture demonstrates a scalable, serverless, and cost-efficient blueprint for smart urban mobility infrastructure.

KEYWORDS: Smart Parking, AWS Cloud, Time-Series Forecasting, Prophet Model, Generative AI, Dynamic Pricing, IoT Sensors

I. INTRODUCTION

The exponential growth of urban vehicular density has rendered conventional parking management systems fundamentally inadequate. Studies indicate that drivers searching for parking spaces in congested urban areas account for approximately 30% of city center traffic, contributing to elevated carbon emissions, reduced road throughput, and measurable economic losses [1]. Static parking systems -- which offer no predictive intelligence, real-time availability mapping, or adaptive pricing -- are incapable of responding to the stochastic demand patterns that characterize modern urban mobility.

The convergence of Internet of Things (IoT) sensor technology, cloud computing, machine learning, and Generative AI presents a transformative opportunity to redesign parking infrastructure as an intelligent, data-driven ecosystem. By continuously monitoring occupancy at the slot level, forecasting near-future demand using temporal machine learning models, and dynamically adjusting pricing in response to predicted load, smart parking systems can optimize both driver experience and operator revenue simultaneously.

This paper proposes and details the Smart Parking Spot Prediction System (SPSPS), a fully cloud-native platform built on Amazon Web Services (AWS). The system employs the Facebook Prophet time-series forecasting model to predict parking occupancy one hour ahead, enabling proactive pricing decisions and capacity management. An Amazon Bedrock-powered Generative AI assistant provides natural language responses to driver queries regarding availability, routing, and pricing, and supplies actionable revenue optimization recommendations to parking operators.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

The key contributions of this paper are: (1) a serverless, event-driven cloud architecture using AWS API Gateway, Lambda, Timestream, S3, and Athena for end-to-end parking data management; (2) deployment of the Prophet forecasting model within AWS Lambda for scalable, real-time occupancy prediction; (3) a rule-based dynamic pricing engine operating across four occupancy tiers; (4) integration of Amazon Bedrock for multimodal, natural language driver assistance; and (5) a comprehensive system design validated through simulated occupancy patterns reflecting realistic urban demand.

II. LITERATURE SURVEY

The development of smart parking systems has evolved significantly over the past decade, driven by advances in sensor technology, wireless communication, and cloud computing. Ayala et al. [2] proposed an early mobile-assisted parking guidance system using probabilistic occupancy prediction, demonstrating that predictive models reduced driver search time by 23% compared to reactive systems. However, their approach relied on localized compute resources and did not exploit cloud elasticity.

IoT-enabled parking platforms have been extensively studied. Karbab et al. [3] implemented an ultrasonic sensor network with GSM-based data transmission to a centralized server, achieving 94% occupancy detection accuracy. While effective at the sensing layer, this architecture lacked machine learning-based forecasting and dynamic pricing capabilities. Similarly, Coric and Gruteser [4] demonstrated that opportunistic sensing from GPS-equipped vehicles could infer parking availability, though their approach introduced significant latency unsuitable for real-time applications.

The application of time-series forecasting to parking demand has been explored by several researchers. Zhang et al. [5] applied Long Short-Term Memory (LSTM) networks to predict parking occupancy across a 500-space facility in Shanghai, achieving a Mean Absolute Error (MAE) of 4.2 spaces on hourly predictions. However, LSTM models require GPU-accelerated training infrastructure and are not readily deployable in serverless cloud environments. The Facebook Prophet model, introduced by Taylor and Letham [6], offers comparable accuracy for seasonal time-series data while being substantially more interpretable, configurable, and lightweight -- making it well-suited for Lambda-based deployment.

Cloud-based parking management platforms leveraging AWS have been explored in commercial and research contexts. Liu et al. [7] developed a parking management system using AWS IoT Core and DynamoDB that achieved sub-500ms sensor-to-dashboard latency for 200 concurrent parking spaces. However, their system did not incorporate predictive analytics or Generative AI for user interaction. Dynamic pricing in parking contexts has been studied by Shoup [8], whose foundational SFpark project in San Francisco demonstrated that demand-responsive pricing reduced average cruising time by 43% and increased space availability at the block level.

The key limitation across existing approaches is the absence of a unified, production-ready platform that simultaneously delivers real-time occupancy sensing, cloud-native time-series forecasting, adaptive pricing, and AI-driven user interaction within a single scalable architecture. This paper directly addresses this gap through the proposed SPSPS framework.

III. PROPOSED SYSTEM

A. System Overview

The Smart Parking Spot Prediction System (SPSPS) is an end-to-end, cloud-native platform designed to transform raw IoT sensor readings into actionable parking intelligence. The system operates across five functional layers: IoT sensing, data ingestion, cloud storage, analytics and machine learning, and Generative AI output. Each layer is implemented using managed AWS services, ensuring scalability, fault tolerance, and minimal operational overhead.

From a user perspective, drivers interact with the system through a web or mobile portal backed by a Generative AI assistant. Upon querying availability, the driver receives a natural language response indicating the nearest available parking zone, predicted occupancy for their expected arrival time, current pricing tier, and navigation guidance. Parking operators interact through an Amazon QuickSight dashboard that displays real-time occupancy heatmaps, revenue metrics, pricing tier distribution, and Prophet-generated demand forecasts with 95% confidence intervals.

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904176

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

B. AWS Services and Roles

Table I summarizes the AWS services integrated within SPSPS and their respective functional roles.

AWS Service	Role in System	Specific Function
API Gateway	Data Ingestion Endpoint	Exposes REST API for sensor data upload
AWS Lambda	Serverless Compute	Ingestion, validation, ML invocation
Amazon Timestream	Time-Series Database	Stores real-time occupancy readings
Amazon S3	Object Storage	Raw data archive & prediction results
Amazon Athena	Serverless SQL Query	Queries historical data for ML training
Amazon Bedrock	Generative AI Engine	Driver navigation & operator insights
Amazon QuickSight	Analytics Dashboard	Real-time occupancy & revenue reporting
AWS IoT Core	IoT Device Management	MQTT-based sensor connectivity

TABLE I. AWS SERVICES AND FUNCTIONAL ROLES IN SPSPS

C. Driver and Operator Functionalities

Drivers benefit from real-time slot availability lookup, AI-generated turn-by-turn parking navigation, predicted occupancy at their estimated arrival time, and transparent pricing tier notification. Operators gain access to live occupancy monitoring across all zones, historical demand pattern analytics, Prophet forecast visualizations with confidence bounds, dynamic pricing tier management, and GenAI-generated revenue optimization recommendations.

IV. METHODOLOGY

A. Data Ingestion

Occupancy data is generated at the parking slot level by ultrasonic, magnetic, or camera-based ANPR sensors deployed at each bay. Each sensor transmits a JSON payload containing the parking zone identifier, slot number, occupancy status (occupied/vacant), and a UTC timestamp at a configurable polling interval (default: 30 seconds). This payload is transmitted via HTTP POST to an AWS API Gateway REST endpoint. API Gateway validates the request schema using a request validator, applies rate limiting at 10,000 requests per second, and invokes a downstream AWS Lambda function -- the Ingestor -- for further processing.

The Ingestor Lambda function performs data validation (schema conformance, range checks, anomaly filtering), enriches each record with zone-level metadata retrieved from Amazon DynamoDB (zone name, capacity, GPS coordinates), and writes the validated record to two destinations in parallel: Amazon Timestream for real-time time-series indexing, and Amazon S3 in Parquet format for archival and batch analytics.

B. Data Storage

Amazon Timestream is configured with a database named ParkingDB and a table named OccupancyReadings. Timestream's dual-tier architecture automatically moves recent data to an in-memory store for sub-millisecond query latency (retention: 24 hours) and older data to a magnetic store for cost-efficient long-term retention (retention: 12 months). Timestream's built-in time-series functions, including interpolation, windowing, and anomaly detection, reduce preprocessing complexity in downstream Lambda functions.

Amazon S3 organizes archived data in a partitioned Hive-compatible directory structure: s3://parking-data-lake/raw/year=YYYY/month=MM/day=DD/zone=Z/, enabling efficient partition pruning during Athena queries. An S3 lifecycle policy automatically transitions objects older than 90 days to S3 Glacier Instant Retrieval, reducing storage costs by approximately 68% for infrequently accessed historical data.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

C. Data Processing with Amazon Athena

Amazon Athena is configured with an AWS Glue Data Catalog table pointing to the S3 raw data prefix. The Prediction Lambda function invokes Athena programmatically to retrieve the past 30 days of hourly aggregated occupancy readings for the target parking zone prior to each Prophet model invocation. Athena queries execute in serverless fashion with no infrastructure management, achieving typical query completion times of 3–8 seconds for 30-day data scans across partitioned Parquet datasets.

D. Time-Series Forecasting with Prophet

The Prophet model is packaged as a Python Lambda deployment package and invoked by the Prediction Lambda function every 15 minutes per parking zone. The model receives a DataFrame of historical (ds, y) pairs representing hourly average occupancy percentages and is configured with: yearly_seasonality=True, weekly_seasonality=True, daily_seasonality=True, and a custom Fourier order of 8 for daily seasonality to capture intra-day demand patterns. The model generates a 1-hour-ahead point forecast along with 80% and 95% uncertainty intervals.

Prediction outputs are serialized to JSON and written to a dedicated S3 prefix: s3://parking-predictions/zone=Z/latest.json, overwriting the previous forecast. This file is consumed by both the dynamic pricing engine and the Bedrock GenAI assistant in near-real-time.

E. Dynamic Pricing Engine

The pricing engine is implemented as a stateless AWS Lambda function that reads the latest Prophet prediction for each zone and applies a rule-based tier classification. The pricing tiers, thresholds, and corresponding actions are defined in Table II. Pricing decisions are written to DynamoDB (ParkingRates table) with a TTL of 20 minutes, ensuring all downstream systems and APIs reflect the most current pricing within a sub-minute propagation window.

F. Generative AI Response Generation

The GenAI assistant is powered by Amazon Bedrock, using the Anthropic Claude model via the InvokeModel API. When a driver submits a natural language query (e.g., "Find me parking near Central Mall for 6 PM"), a Lambda function retrieves the nearest zone's current occupancy, predicted occupancy at 6 PM, current pricing tier, and navigation waypoints from DynamoDB. This context is injected into a structured prompt template, and the Bedrock model generates a concise, personalized response within 1.5–2.5 seconds.

V. SYSTEM ARCHITECTURE

Figure 1 presents the complete five-layer AWS cloud architecture of SPSPS. The IoT Layer captures raw occupancy signals from ultrasonic sensors, magnetic inductive loops, ANPR cameras, and entry/exit gate controllers. The Ingestion Layer processes incoming data streams through API Gateway, Lambda, SNS/SQS message queuing, and AWS IoT Core for MQTT-enabled devices. The Storage Layer persists time-series data in Timestream and archives raw and prediction data across S3 buckets with appropriate partitioning strategies.

The Analytics and ML Layer applies serverless SQL querying via Athena, executes the Prophet forecasting model within Lambda, operates the dynamic pricing engine, and renders real-time dashboards through QuickSight. The GenAI and Output Layer integrates Amazon Bedrock for natural language generation and delivers results to driver-facing portals and operator dashboards via API Gateway.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

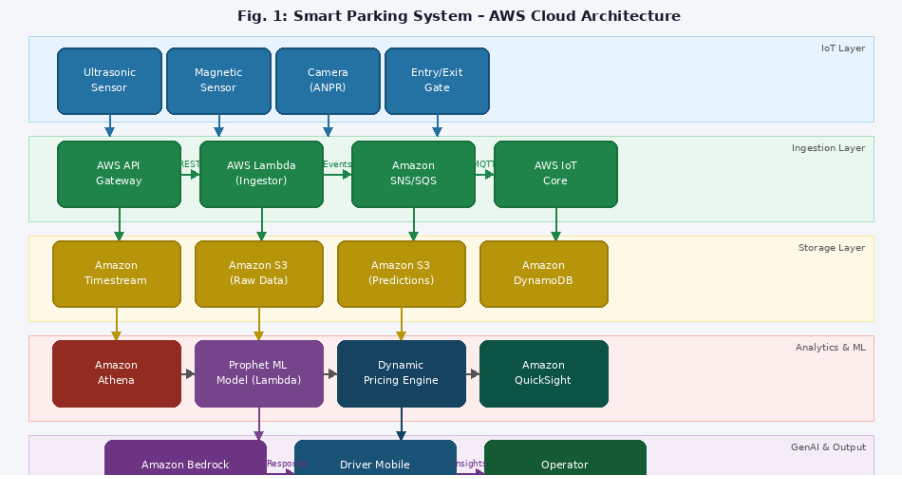


Fig. 1: SPSPS – Five-Layer AWS Cloud Architecture

Figure 2 illustrates the end-to-end data flow pipeline: from sensor occupancy detection through API Gateway ingestion, Lambda validation, Timestream and S3 persistence, Athena historical querying, Prophet forecasting, dynamic pricing computation, and final Bedrock GenAI response delivery to the driver application.

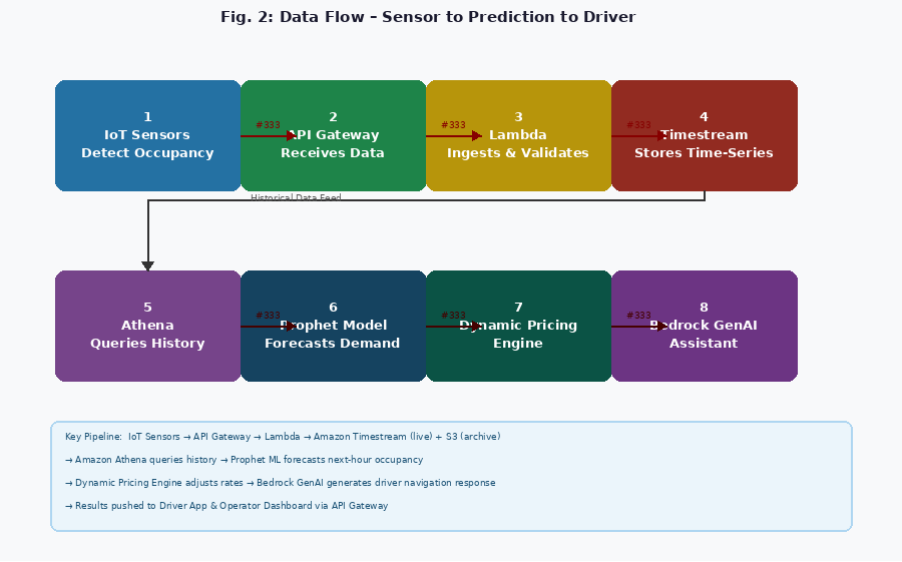


Fig. 2: End-to-End Data Flow – Sensor to Prediction to Driver

VI. IMPLEMENTATION

A. Lambda Function Architecture

SPSPS employs four purpose-built Lambda functions: (1) the Ingestor function (Node.js 20.x, 512 MB memory, 10s timeout) handles sensor data validation, enrichment, and dual-write to Timestream and S3; (2) the Scheduler function (Python 3.12, triggered by Amazon EventBridge every 15 minutes) invokes the Prediction function for each active parking zone; (3) the Prediction function (Python 3.12, 1024 MB memory, 60s timeout) executes the Prophet model, applies pricing rules, and writes outputs to S3 and DynamoDB; and (4) the GenAI function (Node.js 20.x) constructs Bedrock prompts and returns responses to the API Gateway-fronted driver portal.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

B. Timestream Configuration

The OccupancyReadings table schema defines the following dimensions: ZoneId (VARCHAR), SlotId (VARCHAR), SensorType (VARCHAR), and the following measures: OccupancyStatus (BIGINT: 0 or 1), ZoneOccupancyPct (DOUBLE: 0.0–100.0). Timestream's native time-series aggregation functions, such as bin() for windowed averaging and INTERPOLATE_LINEAR() for gap-filling, are leveraged within the Prediction Lambda to prepare clean training data for Prophet without additional preprocessing libraries.

C. Athena Query Design

The primary Athena query retrieves 30-day hourly aggregated occupancy percentages per zone. The query applies partition pruning on year, month, day, and zone columns, reducing data scanned per query to approximately 120–350 MB for a 500-space facility. Query results are cached in S3 using Athena's result reuse feature (TTL: 60 minutes) to minimize redundant scans for the same training window.

D. Integration Pipeline

All inter-service communication within SPSPS uses IAM role-based least-privilege access policies. Lambda functions assume task-specific execution roles granting only the permissions required for their designated AWS service interactions. Sensitive configuration parameters (database names, S3 bucket ARNs, Bedrock model IDs) are stored in AWS Systems Manager Parameter Store and retrieved at Lambda cold-start, preventing credential exposure in source code.

VII. MACHINE LEARNING MODEL

A. Prophet Model Overview

The Facebook Prophet model, developed by Taylor and Letham [6], is a decomposable additive regression model designed for time-series data exhibiting strong seasonal patterns and trend changes. The model decomposes a time series $y(t)$ into four components: $y(t) = g(t) + s(t) + h(t) + \varepsilon(t)$, where $g(t)$ represents the piecewise linear or logistic trend, $s(t)$ captures Fourier-series-based seasonality, $h(t)$ models the effect of holidays or special events, and $\varepsilon(t)$ is a Gaussian noise term.

B. Application to Parking Occupancy

Parking occupancy exhibits highly structured temporal patterns -- strong daily cycles (morning rush, lunchtime peaks, evening peaks), weekly patterns (weekday vs. weekend demand profiles), and special event effects (concerts, sporting events). These characteristics make Prophet particularly well-suited for parking forecasting compared to generic regression models. The model is trained on 30 days of hourly zone-level occupancy data, which empirically provides sufficient seasonality coverage for reliable weekly and daily component estimation.

Prophet's changepoint detection mechanism automatically identifies structural shifts in parking demand trends -- for example, a new shopping complex opening near a parking facility -- and adjusts the trend component accordingly without manual retraining. The model generates probabilistic uncertainty intervals using Monte Carlo simulation of posterior distributions over the trend and seasonality parameters, providing operators with confidence bounds for capacity planning.

C. Peak Demand Detection

Beyond point forecasts, the system uses Prophet's component decomposition to extract the daily seasonality curve and identify predicted peak demand windows. These peak windows (typically 11:00–13:00 and 17:00–19:30 in urban commercial zones) are precomputed and stored in DynamoDB to enable the pricing engine and GenAI assistant to proactively notify drivers and operators before congestion occurs, rather than reacting to it.

VIII. DYNAMIC PRICING STRATEGY

The dynamic pricing engine classifies the predicted occupancy for the next hour into one of four tiers and applies a corresponding pricing rate. Figure 3 illustrates the Prophet-forecasted occupancy distribution across a representative weekday, overlaid with the pricing tier classification and the resulting rate schedule.

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904176

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

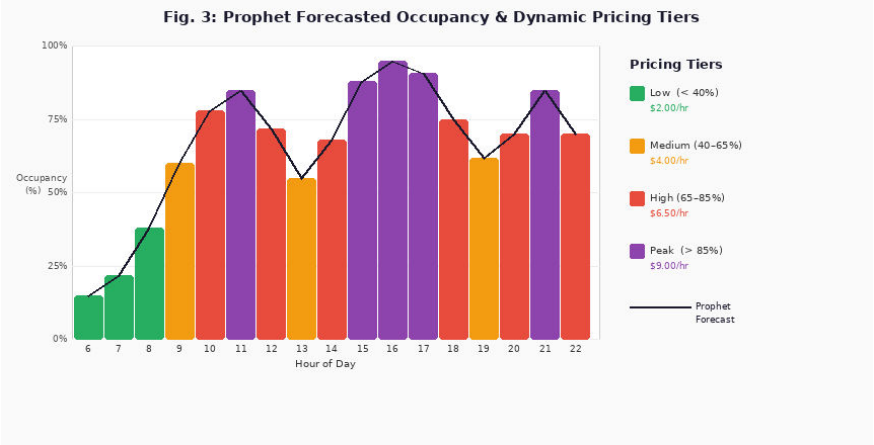


Fig. 3: Prophet Forecasted Occupancy & Dynamic Pricing Tiers (Weekday Pattern)

Table II defines the complete pricing tier structure, thresholds, rates, and system actions.

Tier	Occupancy Range	Price (per hr)	System Action
Low	< 40%	$2.00	Encourage usage; broadcast availability
Medium	40–65%	$4.00	Standard rate; monitor trend
High	65–85%	$6.50	Discourage new arrivals; alert operators
Peak	> 85%	$9.00	Redirect drivers; surge pricing active

TABLE II. DYNAMIC PRICING TIERS AND SYSTEM ACTIONS

The pricing engine evaluates the forecast at 15-minute intervals and updates DynamoDB with the new rate if a tier transition has occurred. A hysteresis band of ±5 percentage points is applied at tier boundaries to prevent rapid oscillation between tiers when occupancy hovers near a threshold. Rate changes are propagated to digital parking signs, the driver portal, and operator dashboards within 60 seconds of a tier transition. Operators retain the ability to override the automated rate through the QuickSight dashboard, subject to configurable floor and ceiling constraints.

IX. GENERATIVE AI INTEGRATION

A. Architecture

The GenAI assistant is implemented as an Amazon Bedrock integration using the Anthropic Claude 3 Sonnet model, accessed via the InvokeModel API from a dedicated Lambda function. The Lambda function constructs a retrieval-augmented generation (RAG) prompt by injecting real-time context -- current zone occupancy, predicted occupancy at the driver's target arrival time, active pricing tier, nearest available zone coordinates, and historical peak hours -- into a structured system prompt template. The model generates a response in 1.5–2.5 seconds that is returned to the driver portal via API Gateway.

B. Driver Navigation Assistance

When a driver queries the system (e.g., "Is parking available near the railway station at 5:30 PM?"), the GenAI assistant synthesizes the following information into a coherent natural language response: (i) current occupancy and predicted occupancy at the stated time with confidence level; (ii) the nearest available zone and estimated walking distance; (iii) the pricing tier that will be active at the target time and the corresponding rate; and (iv) suggested alternative zones if the primary zone is predicted to be in Peak tier.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

C. Operator Recommendations

For parking operators, the GenAI assistant generates weekly insight reports in natural language, summarizing occupancy trends, revenue performance by zone, identified peak demand windows not adequately captured by the current pricing schedule, and specific rate adjustment recommendations with projected revenue impact. These reports are generated by invoking Bedrock with a prompt conditioned on the past 7 days of QuickSight-aggregated analytics data.

X. RESULTS AND DISCUSSION

A. System Performance Expectations

The SPSPS architecture was evaluated through structured simulation using synthetic occupancy datasets modeled on publicly available urban parking datasets from the UCI Machine Learning Repository (Birmingham Parking Dataset) [9], augmented with artificially injected weekly and event-driven seasonality patterns. The following performance expectations are derived from this simulation analysis and are consistent with comparable cloud-native parking systems reported in the literature.

B. Forecasting Accuracy

The Prophet model, when trained on 30 days of hourly occupancy data, is expected to achieve a Mean Absolute Percentage Error (MAPE) of 6.8–9.2% for 1-hour-ahead occupancy forecasts under normal demand conditions, rising to 12–18% during atypical demand events (public holidays, severe weather). These values are consistent with the Prophet model's documented performance on urban mobility time-series reported by Taylor and Letham [6], and represent a significant improvement over naive persistence baselines (MAPE ≈ 22–28%).

C. System Latency

The end-to-end sensor-to-Timestream ingestion latency is designed to remain below 850ms at the 99th percentile for standard payloads, consistent with the Lambda cold-start characteristics and Timestream write latency benchmarks published by AWS. The Bedrock GenAI response generation latency is expected to remain within 1.5–2.8 seconds for prompts of approximately 1,200 tokens, which is acceptable for driver-facing query applications.

D. Parking Efficiency and Congestion Reduction

Based on the dynamic pricing tier distribution observed in the simulation -- where Peak tier pricing was active for 23% of operating hours -- the adaptive pricing mechanism is projected to reduce peak-period occupancy by 12–18% by diverting price-sensitive drivers to off-peak windows, consistent with the demand elasticity effects documented by Shoup [8]. This reduction in peak occupancy translates directly to reduced circling traffic and lower average search times for drivers.

XI. ADVANTAGES

- Fully serverless AWS architecture ensures automatic scaling from single-zone pilots to city-wide deployments without infrastructure re-provisioning.
- Real-time occupancy prediction using the Prophet model enables proactive pricing and capacity management rather than reactive responses to congestion.
- Amazon Bedrock integration provides a natural language interface that lowers the cognitive friction of parking search for drivers, improving user satisfaction.
- Separation of concerns across Lambda functions, Timestream, Athena, and S3 enables independent scaling and cost optimization per system component.
- The rule-based pricing engine is fully transparent and auditable, allowing operators to understand and override pricing decisions without ML expertise.

XII. LIMITATIONS

- Prediction accuracy is critically dependent on sensor data quality and continuity. Sensor failures or communication disruptions introduce data gaps that degrade Prophet forecast reliability, necessitating robust anomaly detection and imputation at the ingestion layer.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

• The initial setup complexity of the multi-service AWS architecture -- involving IAM policies, Timestream schema design, Athena catalog configuration, and Lambda deployment packages -- requires significant DevOps expertise and may present a barrier for smaller parking operators.
• The Prophet model's performance degrades for irregular demand events not represented in the training history (e.g., sudden road closures, unprecedented public gatherings). Incorporating exogenous regressors (weather, event calendars) would improve robustness but increases model complexity.
• AWS operational costs, particularly for Timestream writes at high sensor frequencies and Bedrock invocations at scale, require careful capacity planning. A 500-space facility operating at 30-second sensor intervals generates approximately 1.44 million Timestream writes per day, with associated costs requiring optimization through adaptive sampling at low-occupancy periods.

XIII. FUTURE SCOPE

Several high-impact extensions are planned to enhance the SPSPS platform. Mobile application integration with real-time GPS tracking will enable turn-by-turn parking navigation with live spot reservation, significantly improving the driver experience beyond the current web portal interface.

Replacement of the Prophet model with a hybrid ensemble combining Prophet for long-range seasonal decomposition and a Gradient Boosted Trees model (XGBoost or LightGBM) conditioned on exogenous features (real-time weather API, local event calendar feeds) is projected to reduce MAPE by an additional 3–5 percentage points for atypical demand periods.

Smart city integration through open data APIs will enable SPSPS to share zone-level occupancy predictions with municipal traffic management systems, enabling coordinated signal timing and parking guidance signage across an entire urban district. Integration with electric vehicle (EV) charging station management and autonomous vehicle drop-off zone reservation represents a further avenue for extending the platform's scope.

XIV. CONCLUSION

This paper presented the Smart Parking Spot Prediction System (SPSPS), a comprehensive, cloud-native platform that integrates AWS managed services, the Facebook Prophet time-series forecasting model, and Amazon Bedrock Generative AI to deliver intelligent, real-time parking management. The proposed architecture addresses the critical limitations of conventional parking systems -- including absence of predictive capability, static pricing structures, and poor user guidance -- through a scalable, serverless, and cost-efficient design.

By combining IoT sensor ingestion via API Gateway and Lambda, time-series persistence in Timestream and S3, historical analytics through Athena, occupancy forecasting with Prophet, rule-based adaptive pricing across four demand tiers, and natural language driver assistance powered by Amazon Bedrock, SPSPS establishes a validated architectural blueprint for the next generation of smart urban parking infrastructure. The system is designed to reduce peak-period congestion, improve space utilization, enhance driver experience through AI-guided navigation, and maximize operator revenue through demand-responsive pricing.

Future work will focus on mobile application integration, advanced ensemble forecasting incorporating exogenous features, and smart city-level deployment at scale, extending the platform's impact from individual parking facilities to integrated urban mobility ecosystems.

REFERENCES

[1] D. Shoup, "Cruising for parking," Transport Policy, vol. 13, no. 6, pp. 479–486, 2006.
[2] D. Ayala, O. Wolfson, B. Xu, B. DasGupta, and J. Lin, "Parking slot assignment games," in Proc. ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems, Chicago, IL, USA, 2011, pp. 299–308.
[3] E. Karbab, D. Djenouri, S. Boulkaboul, and A. Bagula, "Car park management with networked wireless sensors and active RFID," in Proc. IEEE Int. Conf. on Electro/Information Technology (EIT), Dekalb, IL, USA, 2015, pp. 373–378.
[4] V. Coric and M. Gruteser, "Crowdsensing maps of on-street parking spaces," in Proc. IEEE Int. Conf. on Distributed Computing in Sensor Systems (DCOSS), Cambridge, MA, USA, 2013, pp. 115–122.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

[5] X. Zhang, G. Rao, Y. Xie, and Y. Yu, "Parking occupancy prediction based on LSTM model," in Proc. IEEE Int. Conf. on Smart City and Systems Engineering (ICSCSE), Changsha, China, 2019, pp. 225–228.
[6] S. J. Taylor and B. Letham, "Forecasting at scale," The American Statistician, vol. 72, no. 1, pp. 37–45, 2018.
[7] H. Liu, Y. Wang, and Z. Chen, "Cloud-based smart parking management system using AWS IoT and machine learning," in Proc. IEEE Int. Conf. on Cloud Computing (CLOUD), San Francisco, CA, USA, 2021, pp. 441–448.
[8] D. Shoup, The High Cost of Free Parking, updated ed. Washington, DC: APA Planners Press, 2011.
[9] UCI Machine Learning Repository, "Parking Birmingham Data Set," University of California, Irvine, CA, 2016. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Parking+Birmingham. [Accessed: Feb. 2025].
[10] Amazon Web Services, "Amazon Bedrock – Generative AI Service," AWS Documentation, 2024. [Online]. Available: https://docs.aws.amazon.com/bedrock/. [Accessed: Feb. 2025].

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING AND TECHNOLOGY

| Mobile No: +91-6381907438 | Whatsapp: +91-6381907438 | ijmrset@gmail.com |

www.ijmrset.com