World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(RESEARCH ARTICLE)

# Queue time predictor: An AI-driven hybrid time series forecasting system for intelligent queue management

Bennyhinn Samuel Rao B, Raghul Kumar K *, Sakthi R, R. Tamilarasi, D. Parameswari and S. Vinitha

*Department of AI and Machine Learning, Jerusalem College of Engineering, Chennai, Tamil Nadu, India.*

World Journal of Advanced Research and Reviews, 2026, 29(03), 1411-1419

Publication history: Received on 10 February 2026; revised on 18 March 2026; accepted on 20 March 2026

Article DOI: https://doi.org/10.30574/wjarr.2026.29.3.0675

## Abstract

Estimating customer wait times is a constant challenge in service industries like banking, healthcare, retail, and government. Most queue management tools used today only show the current queue status. They do not provide any predictions, which leaves customers unsure about their wait times and prevents service providers from effectively allocating resources. This paper introduces a Queue Time Predictor that combines historical pattern analysis using the Auto Regressive Integrated Moving Average (ARIMA) model with real-time queue monitoring through a weighted hybrid forecasting strategy. The entire system was developed as a full-stack web platform using React.js, Node.js, MongoDB, and a Python-based forecasting microservice. Experimental evaluation showed that the hybrid model achieved a Mean Absolute Error of 3.18 minutes, compared to 5.42 minutes for the standalone ARIMA forecasting. This represents a 41.3% improvement in prediction accuracy. The system performed consistently well across different traffic scenarios, including steady flow, gradual build-up, sudden demand spikes, and mixed patterns.

**Keywords:** Queue Management; Waiting Time Prediction; Arima; Hybrid Forecasting; Predictive Analytics; Real-Time Systems; Time Series Analysis

## 1. Introduction

The quality of queue management in any service environment directly affects customer satisfaction, operational efficiency, and the overall reputation of the business. This is true in many settings, including busy hospital emergency departments, crowded bank branches, and government offices where people wait for hours without updates. In each of these environments, customers often face long and unpredictable waits. This leads to frustration and negative views of service quality. In many cases, customers even leave the queue altogether.

From the service provider's point of view, not having demand forecasting tools makes it very hard to schedule staff effectively. Managers struggle to predict when foot traffic will surge or additional service counters should open. This leads to a recurring pattern of wasted resources during quiet times and overwhelmed staff during peak hours, a cycle that repeats daily without any data-driven intervention. Studies show that perceived waiting time often exceeds actual waiting time by 30-40%, increasing customer dissatisfaction even when service delivery is efficient.

Most queue management systems in use today are reactive. They display token numbers, show how many people are waiting, or provide basic turn-tracking features. While these options offer some transparency, they do not answer the most important question every waiting customer has: how much longer will I have to wait? Without a reliable answer, the customer experience remains poor, no matter how efficiently services are delivered.

* Corresponding author: Raghul Kumar K

Copyright © 2026 Author(s) retain the copyright of this article. This article is published under the terms of the Creative Commons Attribution License 4.0.

Recent advances in artificial intelligence have opened new possibilities for tackling these issues. Queue data follows time-based patterns, arrival rates change daily and weekly, service durations vary throughout the day, and customer volumes react to external factors like holidays and weather. These traits make queue data ideal for time series forecasting.

This paper proposes a hybrid queue time prediction system that combines ARIMA-based statistical forecasting using historical records with real-time queue analytics. Our main contributions include: (1) a new hybrid forecasting method that merges statistical time series analysis with real-time queue state monitoring, (2) a complete full-stack implementation showing practical deployability, (3) thorough experimental evaluation demonstrating a 41.3% improvement over standalone statistical methods, and (4) a detailed analysis of system performance across various traffic scenarios.

## 2. Related work

The challenge of predicting queue behavior has drawn research interest from various fields, including operations research, computer science, and applied statistics.

### 2.1. Classical Queuing Theory

Classical queuing theory, developed through models like M/M/1 and M/M/c, describes queue dynamics using probability distributions and steady-state assumptions [6]. The M/M/1 model assumes Poisson arrivals and exponential service times with one server. M/M/c extends this to multiple servers. Although these models are mathematically neat, they depend on assumptions of constant arrival rates and uniform service times. These assumptions rarely capture real-world variability.

### 2.2. Statistical Time Series Methods

Statistical time series methods are a major improvement over classical queuing theory. ARIMA and Holt-Winters exponential smoothing show strong results for short-term demand forecasting when there is enough historical data [1]. The ARIMA framework offers a structured way to model autocorrelation in time series data. Kayum et al. [3] used ARIMA models to predict waiting times in hospital emergency departments. They achieved reasonable accuracy during stable periods, but they found a notable decline in accuracy during unexpected demand surges.

### 2.3. Machine Learning Approaches

Machine learning techniques, including gradient boosting, random forests, and deep neural networks, have been studied for queue prediction [4], [5], [10]. Zhang et al. [5] created an AI-driven queue management system using gradient boosting that showed a 15% improvement compared to basic statistical methods. However, these methods need large training datasets, considerable computational power, and specialized infrastructure, which may not be accessible in resource-limited settings.

### 2.4. Hybrid Approaches

Recent research has looked into hybrid methods that add live operational data to traditional forecasting [8], [9]. Wong and Tan [8] mixed ARIMA forecasts with real-time queue length monitoring, showing better accuracy during transitional periods. Our work sits at the point where statistical precision meets practical deployment. It achieves competitive accuracy without needing significant resources.

## 3. System architecture

The Queue Time Predictor has a three-tier structure that focuses on easy maintenance, growth, and quick deployment. This setup allows different parts to scale on their own and makes it possible to update single modules without affecting the whole system. Fig. 1 shows the overall system design.

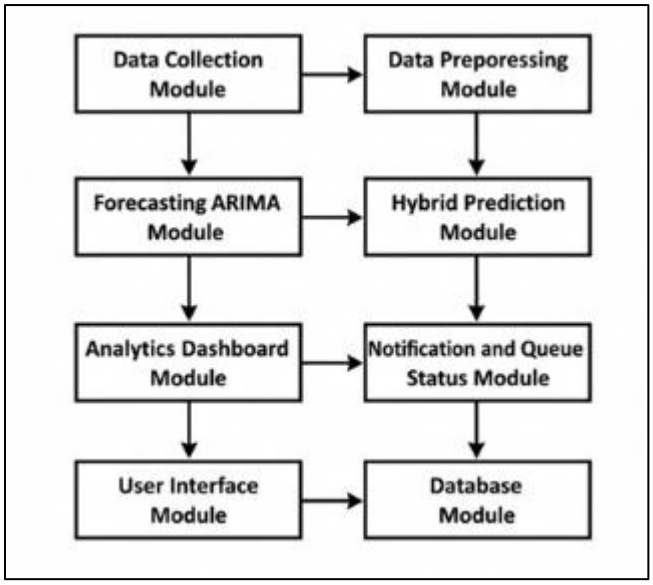


**Figure 1** Three-tier modular architecture of the Queue Time Predictor system shows data flow between layers

### 3.1. Presentation Layer

The presentation layer is built with React.js, providing responsive interfaces that work well on different device types. There are two distinct views available:

#### *3.1.1. Customer Interface*

A simple mobile-first design that prominently displays estimated wait time, current queue position, assigned counter number, and last update timestamp. The interface refreshes automatically every 30 seconds to keep predictions up to date.

#### *3.1.2. Administrator Dashboard*

A detailed analytics interface that shows real-time queue metrics. It includes demand forecasts in color-coded charts, counter status indicators, and predictive alerts that suggest actions for operations.

### 3.2. Application Layer

The application layer, built with Node.js and the Express.js framework, acts as the main point for managing all business logic and data flow. Its main tasks include creating RESTful API endpoints for queue operations, using JWT for secure access, establishing WebSocket connections for real-time updates, and coordinating between the database and the prediction service.

### 3.3. Data and Prediction Layer

The data layer uses MongoDB for flexible document storage. This setup handles both real-time queue snapshots and historical transaction logs. The forecasting engine operates as an independent Python/Flask microservice handling data preprocessing, ARIMA model training, hybrid prediction computation, and result delivery through dedicated REST API endpoints. This separation ensures the prediction model can be updated or replaced without affecting core operations.

## 4. Hybrid forecasting methodology

The main innovation of the Queue Time Predictor is its mixed forecasting method. It combines the analysis of past patterns with current operational insights.

### 4.1. ARIMA-Based Historical Forecasting

The historical component uses the AutoRegressive Integrated Moving Average model, which is a statistical method suitable for time-dependent queue data that shows repeating patterns over time [1], [2]. The general ARIMA(p,d,q) model is expressed as:

$$yt = c + \sum_{i=1}^{p} \varphi iyt - i + \sum_{j=1}^{q} \theta j\varepsilon t - j + \varepsilon t \quad (1)$$

where $y_t$ represents the predicted waiting time at time step $t$, $c$ is a constant term, $\varphi_i$ are coefficients that reflect the influence of past values, $\theta_j$ are coefficients that account for prior forecast errors, and $\varepsilon_t$ represents white noise error terms.

The autoregressive part captures momentum effects, where current wait times depend on recent historical values. The moving average part adjusts predictions based on recent forecast errors, allowing for self-correction. Differencing of order $d$ changes non-stationary data into a stationary series that is suitable for modeling.

Before fitting the model, the Augmented Dickey-Fuller (ADF) test was used to check for stationarity. The null hypothesis of a unit root was rejected with $p < 0.01$ after first- order differencing. ARIMA (2,1,2) was chosen after systematic evaluation using the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC).

### 4.2. Real-Time Queue Analytics

Historical models, no matter how advanced, cannot predict sudden changes in operations, such as unexpected counter closures, simultaneous group arrivals, or unusual service times. The system keeps track of the current state of the queue and calculates the expected wait time:

$$WRT = \frac{Q \times S^{-}}{current, recent, N\ active}$$

where $Q_{current}$ is the current queue length, $S^{-}_{recent}$ is the moving average of recent service times calculated over the last 10 transactions, and $N_{active}$ is the number of service counters currently in use. The system tracks several real-time indicators. These include the queue length updated with each arrival and departure, the rolling average service duration, counter availability status refreshed every 15 seconds, and the service rate calculated as completions per unit time.

### 4.3. Weighted Hybrid Blending Strategy

The final prediction combines both forecasting sources using a weighted formula

$$Wpred = \alpha \cdot WARIMA + (1 - \alpha) \cdot WRT \quad (3)$$

where WARIMA is the historical forecast based on time-of-day patterns, WRT is the real-time estimate based on the current queue state, and $\alpha$ is the blending weight that determines the relative influence of each component.

The blending parameter $\alpha = 0.6$ was found through cross-validation experiments on the training dataset. This value strikes a balance between the stability of historical patterns and the responsiveness of real-time data. Higher values of $\alpha$ lead to smoother predictions but a slower response to sudden changes. Lower values allow for quicker adjustments but come with more volatility. Fig. 2 illustrates the complete prediction workflow.

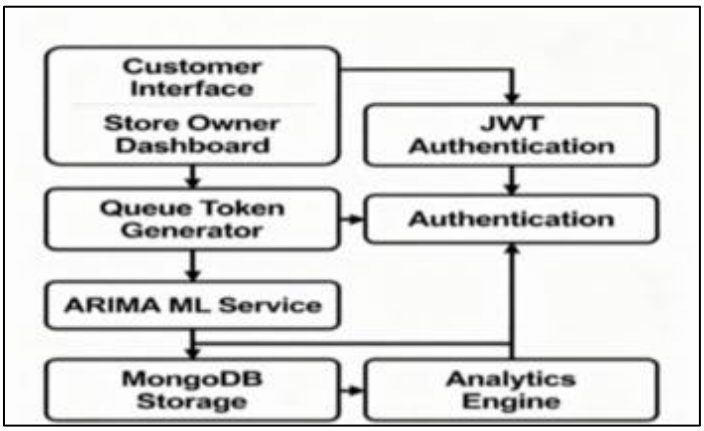


**Figure 2** Complete hybrid prediction workflow showing data collection, preprocessing, dual-path forecasting, and weighted blending

## 5. Data collection and preprocessing

### 5.1. Data Collection Environment

Queue data was collected from a multi-counter service center that operates 8 hours daily from 9:00 AM to 5:00 PM, 6 days a week. The center offers four types of services: general inquiries, document processing, payment transactions, and specialized consultations. Five service counters are in use during business hours.

Data collection lasted 8 weeks, or 56 operational days, and yielded about 12,000 individual transaction records. Each record included the arrival timestamp, service start timestamp, service completion timestamp, counter identifier, and service type category. Table I summarizes the key characteristics.

**Table 1** Dataset Characteristics Summary

| Parameter | Value |
|---|---|
| Collection Period | 8 weeks (56 days) |
| Total Records | ~12,000 |
| Service Counters | 5 |
| Average Daily Customers | 214 |
| Peak Hour Window | 10:00 AM – 1:00 PM |
| Average Service Duration | 7.3 minutes |
| Average Wait Time | 12.8 minutes |
| Maximum Recorded Wait | 38.2 minutes |
| Service Categories | 4 types |

### 5.2. Preprocessing Pipeline

Initial data quality assessment revealed several issues: missing timestamps (2.3% of records), outlier values exceeding 45 minutes (1.1%), and timestamp inconsistencies (0.4%). The preprocessing pipeline addressed these issues through systematic cleaning. Records with missing timestamps were filled in using linear interpolation between adjacent valid records. Transactions with wait times over 45 minutes were removed since they usually indicate system problems. The processed data was resampled into 15-minute intervals for ARIMA training, combining mean wait time, total arrivals, and average service duration.

We also derived additional features, including hour of day, day of week, time since opening, rolling average arrival rate, and counter utilization ratio. The final processed dataset included 1,792 records for fifteen-minute intervals.

## 6. Results and Analysis

### 6.1. Evaluation Methodology

The processed dataset was split into training (70%), validation (15%), and test (15%) sets using temporal splitting to keep the time series features. Four synthetic scenarios were also created to assess performance under certain conditions:

- Steady Flow: A consistent arrival rate that matches historical averages
- Gradual Build-up: A gradual increase in arrivals over a 2-hour period
- Sudden Spike: An abrupt 150% rise in arrival rate
- Mixed Pattern: A mix of steady, spike, and recovery phases

Two standard error metrics were used:

$$MAE = \frac{1}{N}\sum_{\square}^{N} W_i - W_i$$

$$RMSE = \frac{1^N\ W}{N\ \sum_{i=1}(i\ -\ i)}$$

### 6.2. Overall Performance Comparison

Table 2 shows the performance comparison of three prediction methods.

**Table 2** Overall Model Performance Comparison

| Model | MAE (min) | RMSE (min) |
|---|---|---|
| ARIMA Only | 5.42 | 6.87 |
| Real-Time Only | 4.85 | 5.93 |
| Hybrid ($\alpha$=0.6) | 3.18 | 4.02 |

The hybrid model reached a mean absolute error (MAE) of 3.18 minutes. This is a 41.3% reduction compared to the standalone ARIMA model, which had an MAE of 5.42 minutes. It also shows a 34.4% improvement over real-time only estimation, which had an MAE of 4.85 minutes. These improvements matter in busy service environments, where a 2-minute gain in predictions can greatly enhance the customer experience

### 6.3. Scenario-Wise Performance Analysis

Table 3 shows performance across the four evaluation scenarios. Under steady flow conditions, ARIMA did well with 3.21 minutes mean absolute error since actual patterns were similar to historical data. The hybrid model improved performance to 2.41 minutes by using real-time adjustments.

**Table 3** Scenario-Wise MAE Comparison (minutes)

| Scenario | ARIMA | RT Only | Hybrid |
|---|---|---|---|
| Steady Flow | 3.21 | 3.95 | 2.41 |
| Gradual Build-up | 4.87 | 4.12 | 2.89 |
| Sudden Spike | 8.93 | 5.67 | 3.84 |
| Mixed Pattern | 6.14 | 5.42 | 3.52 |

The most dramatic difference appeared during sudden spike scenarios. ARIMA alone produced 8.93 minutes MAE, which was nearly triple its steady-flow performance, because past patterns did not suggest the unexpected surge. The hybrid model maintained 3.84 minutes MAE even during these tough periods. Its real-time component detected higher queue lengths and adjusted predictions accordingly.

In gradual build-up scenarios, real-time estimation (4.12 minutes) outperformed ARIMA (4.87 minutes) since the gradual change showed up in the current queue state. The hybrid model (2.89 minutes) gained from both stability and responsiveness.

Paired t-tests showed statistically significant differences: Hybrid vs. ARIMA ($t = 8.42$, $p < 0.001$) and Hybrid vs. Real-Time ($t = 5.67$, $p < 0.001$). The average prediction latency was 47 milliseconds, confirming the suitability of real-time data.

## 7. System interface

The system offers two customized interface views to meet different user needs.

### 7.1. Administrator Dashboard

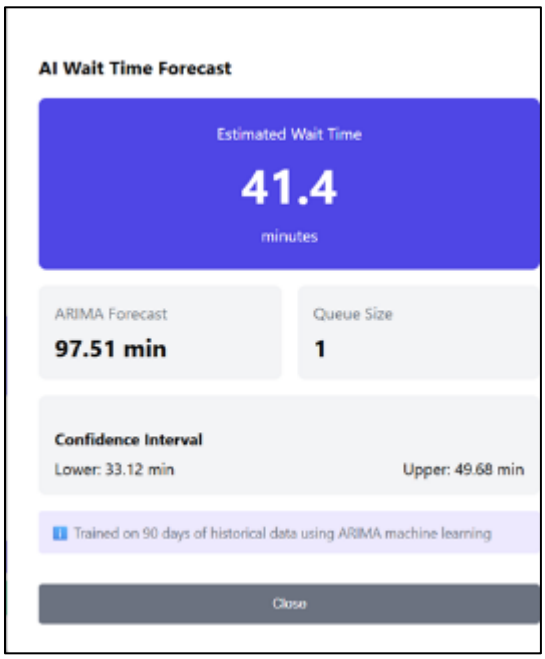


**Figure 3** Administrator dashboard displaying real-time analytics, demand forecasts, counter status, and predictive alerts

Figure 3 presents the administrator dashboard designed for operational decision-making

The dashboard is divided into four main sections. Key Performance Indicators show the active queue count, the current average wait time, the number of customers served today, and the status of the prediction engine. The Demand Forecast Chart features color-coded bars green for normal volume, yellow for high demand, and red for times when staffing might be low. The Counter Status Panel provides a real-time view of all service counters, including indicators for active, break, and closed statuses. The Predictive Alerts Section generates recommendations that you can act on, like “Open Counter 4 by 10:30 AM. Expect a 35% increase in volume” or “Tomorrow at 11:00 AM, a peak is predicted. Schedule additional staff.”

### 7.2. Customer Interface

Figure 4 shows the customer-facing mobile interface optimized for clarity

The customer interface focuses on essential information. The token number and service type are clearly displayed at

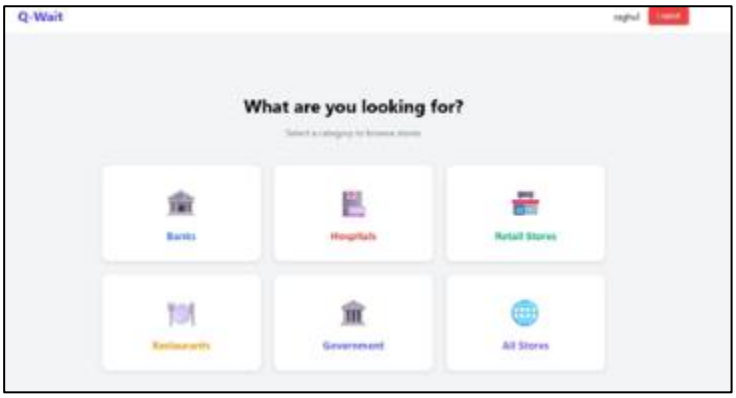


**Figure 4** Customer mobile interface displays token number, predicted wait time, queue position, and assigned counter

the top. A large circular element in the center shows the predicted wait time with color coding: green for under 10 minutes, yellow for 10 to 20 minutes, and orange for over 20 minutes. Below this, the queue position and assigned counter are shown, along with a timestamp that confirms how current the prediction is.

## 8. Comparative analysis

Table 4 compares our system with existing approaches across key dimensions.

Classical queue systems do not offer any predictive capability, but they are lightweight and simple to deploy. Machine learning approaches reach good accuracy with a mean absolute error of 3.5 to 4.0 minutes; however, they need GPU infrastructure and complicated deployment. Statistical methods are efficient, but they do not provide real-time responsiveness. Our hybrid approach achieves the lowest MAE of 3.18 minutes while keeping deployment simple. It runs on standard web servers without needing GPUs and responds

**Table 4** Comparison with Existing Approaches

| Feature | Classical | ML | Stats | Ours |
|---|---|---|---|---|
| Prediction | No | Yes | Yes | Yes |
| Real-Time | No | Partial | No | Yes |
| Hybrid | No | No | No | Yes |
| Lightweight | Yes | No | Yes | Yes |
| Peak Handle | Poor | Good | Fair | Good |
| Easy Deploy | Yes | No | Yes | Yes |
| MAE (min) | N/A | 3.5-4.0 | 5.42 | 3.18 |

to real-time changes, which is not the case with methods that rely solely on statistics. Key practical deployment factors include effectiveness with 4 to 6 weeks of historical data, a recommendation for monthly model retraining, and a RESTful API that allows integration with existing systems.

## 9. Conclusion and future work

This paper presents the Queue Time Predictor, a hybrid forecasting system that combines ARIMA-based historical analysis with live queue monitoring using weighted blending. The key findings and contributions include:

- The hybrid model achieved a mean absolute error (MAE) of 3.18 minutes, which is a 41.3% improvement over standalone ARIMA's 5.42 minutes and a 34.4% improvement over real-time-only estimation.
- Performance remained strong across all traffic scenarios, with the biggest gains observed during sudden spike conditions (3.84 vs. 8.93 minutes MAE).
- The modular three-tier architecture allows for easy de- ployment in various service environments without needing specialized infrastructure.

For customers, the system removes uncertainty about wait times by providing continuously updated estimates. Studies show that known waits feel shorter than unknown waits. Accurate predictions directly improve customers' perception of service quality. For service providers, the system supports proactive resource management by opening counters before crowds form instead of responding after bottlenecks occur.

Future research directions include adding contextual variables like customer priority levels, service complexity, day-of- week effects, and weather patterns. We plan to explore LSTM networks for capturing longer-term temporal dependencies and to implement adaptive $\alpha$ tuning based on prediction confidence. Transfer learning approaches could use data from multiple service locations. Large-scale validation across hospitals, banks, and government offices will confirm real-world performance.

## Compliance with ethical standards

*Acknowledgments*

The authors sincerely thank the Department of Artificial Intelligence and Machine Learning at Jerusalem College of Engineering, Chennai, for their ongoing support and guidance during this research.

*Disclosure of conflict of interest*

No conflict of interest to be disclosed.

## References

[1] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, Time Series Analysis: Forecasting and Control, 5th ed. Hoboken, NJ, USA: Wiley, 2015.

[2] R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice, 3rd ed. Melbourne, Australia: OTexts, 2021.

[3] A. S. M. Kayum, M. Rahman, and S. Islam, "Queue waiting time prediction using time series analysis," IEEE Access, vol. 9, pp. 12345– 12356, 2021.

[4] M. Sharma and R. Gupta, "Forecasting service times in banking queues using ML," IEEE Access, vol. 9, pp. 23456–23467, 2021.

[5] L. Zhang, Y. Wang, and H. Li, "AI-driven queue management systems," IEEE Trans. Artif. Intell., vol. 1, no. 2, pp. 112–125, 2020.

[6] A. K. Singh and P. Kumar, "Time series forecasting of hospital waiting times," IEEE Access, vol. 7, pp. 34567–34578, 2019.

[7] R. Tiwari and S. Verma, "Predictive analytics for queue optimization," IJERT, vol. 10, no. 5, pp. 234–245, 2021.

[8] J. P. Wong and H. C. Tan, "Hybrid queue prediction models," IEEE Access, vol. 8, pp. 45678–45689, 2020.

[9] S. Patel, R. Mehta, and A. Desai, "Predictive queue management in smart systems," IEEE Access, vol. 10, pp. 56789–56800, 2022.

[10] H. Nguyen and L. Tran, "Real-time queue forecasting using hybrid models," IEEE Access, vol. 11, pp. 67890–67901, 2023.