WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

DineFlow: An Offline LAN-Based Restaurant Management System with Machine Learning-Driven Predictive Analytics Final Result Paper

Prof. Arpit Chaudhari [a], Arth Gundapure [a], Harshada Tijare [a], Jiva Bhujade [a], Vinant Rathode [a]

[a] *Department of Computer Science and Engineering, Prof. Ram Meghe Institute of Technology & Research*

A B S T R A C T

Managing a restaurant business can be very extensive, especially when there is no organised system and everything is done manually. Even if a virtual solution is used, there are still some factors that can affect the performance, such as dependency on cloud services. That is why the purpose of this research is to propose a solution that is not only offline-first and standalone but also provides efficiency in handling different restaurant tasks in real time. DineFlow is an offline LAN-based restaurant management system that is built on Python and Flask and PostgreSQL is used as the primary relational database. Machine learning algorithms like Random Forest are used for predictive analytics over historical data and inventory intelligence to increase efficiency of the system . Therefore the goal is to provide an offline alternative to internet-based management systems for better management and execution.

Keywords: Offline-first system, LAN, Restaurant management, Flask, PostgreSQL, Random Forest, Demand forecasting, Role-based access control

1. Introduction

In the restaurant business there are many requirements that can contribute significantly for business growth such as accurate coordination , efficient execution of orders within the required time constraints etc . But along with this, there are several factors that often affect the operability of restaurants . Communication between staff, billing, and fast delivery can affect the execution of orders in any restaurant [1]. Handling them manually can be a prominent challenge for small to large restaurant owners.

Therefore the purpose of this research is to identify such challenges and find online alternatives to resolve them in an efficient manner .In modern restaurant operations multiple interconnected activities such as order processing, table management, kitchen coordination, billing, and management of sales information are involved . The goal is to not only make those tasks possible but to do them in such a manner that staff and owner of restaurant do not face any problem in maintaining data and operating in usual manner

However, the available systems work on cloud-based architecture for storing data and other operations [2] . In situations like internet cutoffs, this can be an issue; thus, the need for a complete offline standalone system that can not only collect data but also manage restaurant activities effectively arose. The proposed system also ensures that reports and analytics , along with real-time updates and tracking , could support restaurant management and decision-making .

2. Problem Statement

The main problem in the system that this research is looking forward to for restaurant management can be internet dependency, as the system shall be in efficient condition all the time, we cannot simply depend on the internet all the time, this can lead to some problems if connectivity is lost [2].

Another problem that can arise can be fragmented responses. As a restaurant involves multiple operations and processes such as Order processing, table management, kitchen coordination, billing, and reporting, there can be different workflows or modules, but we need a centralized system to integrate all of those workflows for efficient coordination.

Not only this, but there can also be some limitations with analytics, as only recording the transactions will not assist the manager with adequate strategy to engage customers, optimize the menu, or any other inventory or restaurant related task .[3].

Research Objectives

1. *Offline LAN Architecture:* This research looks forward to developing a system that has a standalone architecture and almost zero internet dependency but performs operations uninterruptedly.

2. *Integrated Restaurant Management:* To centralize different operational workflows effectively so that the execution can be coordinated and efficient and there is no need for maintaining records of isolated operations.
3. *Secure Role Based Operations:* Implementing role based access control. Different roles can be given access according to their need of operation and the security remains intact.
4. *Real Time Operations and Analytics:* The communication between different sections of the restaurant can happen on a real time basis to avoid the problems that arise in the restaurant due to unnecessary delays and also maintaining records of the historical transactions for better analysis.
5. *Predictive Enhancement:* Implementing Machine learning algorithms to forecast the demand of specific items on the basis of historical data and also perform predictive analysis on the revenue forecasting, possible growth, or challenges for a restaurant.

3. Literature Review

Vinodhini et al. introduced a wireless system for receiving and transmitting orders directly from the table to the kitchen, eliminating the traditional order-taking procedure in restaurants. However, such a system still depends heavily on centralized internet connectivity, keeping connectivity problems at bay [1]. Kleppmann et al. introduced the idea of an offline-first system that tries to eliminate dependency on cloud-based systems, which heavily inspires DineFlow's LAN-based approach toward restaurant management [2]. Tanizaki et al. introduces the idea of using POS data from restaurants to generate demand forecasts using algorithms like Random Forest [3]. However, real time communication is still a concept that needs to be applied for effective order execution. Fette and Melnikov provided an appropriate solution by introducing the idea of WebSocket, which directly contributes to real-time communication between different sections of the system [4]. Schmidt et al. displayed the idea that complex models do not necessarily perform better in the case of demand forecasting, especially when dealing with real-world data . The results were derived by comparing different models on the same dataset and choosing the best possible model based on the results [5]. Breiman explained Random Forest in detail, describing fundamental approaches to effectively implementing ensemble learning and combining multiple decision trees to derive accurate and realistic predictions [6]. Pedregosa et al. is the introduction document of the scikit-learn Python library that consists of open-source implementations of machine learning algorithms, which gave direction to the ML architecture of DineFlow [7]. Chai and Draxler discussed the importance of MAE and RMSE and argued that both should be used together to optimize the evaluation [8]. Ferraiolo and Kuhn introduced the major concept of Role Based Access Control , that is necessary to optimize the security and authentication of systems by restricting access to different roles for respective operations [9].

Table 1 - Summary of related work

Work	Main focus	Observed limitation	Implication
Vinodhini et al. [1]	Wireless touch-screen restaurant ordering	Not offline/LAN-based; internet dependency not addressed	Supports need for a fully offline coordination layer
Kleppmann et al. [2]	Local-first software design principles	Not restaurant-specific; generic software architecture	Motivates offline, LAN-based system design
Tanizaki et al. [3]	Random Forest demand forecasting for restaurant inventory	Accuracy highly dependent on external data (weather, events)	Supports use of RF regression; suggests dataset richness affects results
Fette and Melnikov [4]	WebSocket protocol for real-time communication	Generic protocol, not restaurant-domain specific	Provides technical basis for real-time KDS updates
Schmidt et al. [5]	Comparison of ML models for restaurant sales forecasting	Simple models often matched complex ones; dataset size a factor	Justifies careful evaluation of RF performance against baseline
Breiman [6]	Random Forest algorithm (foundational)	Not domain-specific; general-purpose ensemble method	Provides theoretical basis for chosen ML algorithm
Pedregosa et al. [7]	Scikit-learn ML library	Generic toolkit, not tailored to restaurant use case	Enables local, offline execution of ML model
Chai and Draxler [8]	MAE vs. RMSE as evaluation metrics	Metric choice alone doesn't address low R^2 problem	Supports reporting MAE, RMSE and R^2 together for honest evaluation
Ferraiolo and Kuhn [9]	Role-Based Access Control (foundational)	Generic model, not restaurant-role specific	Supports RBAC implementation for Admin/Manager/Waiter/Cashier/Chef roles

3.1 Proposed System Architecture and Functional Modules

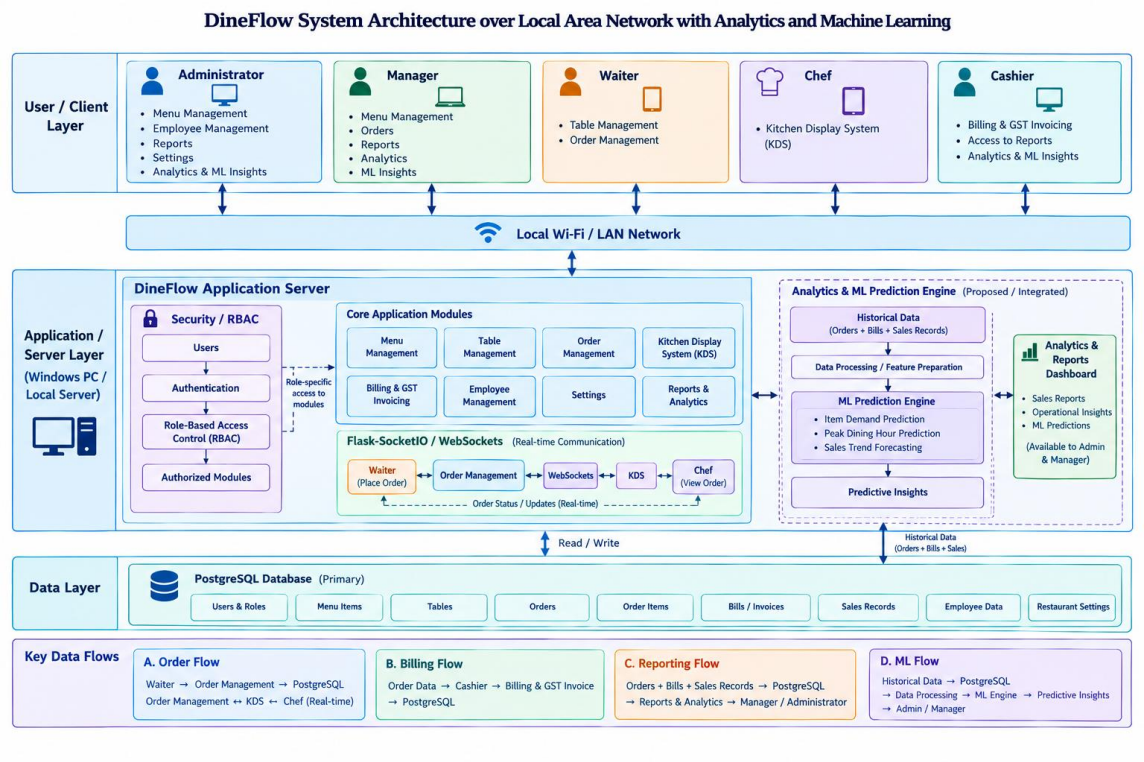


Fig. 1 - DineFlow System Architecture over Local Area Network with Analytics and Machine Learning

3.2 Overview

The proposed solution DineFlow is a centralized architecture based stand alone web application that is a substitute to traditional restaurant and cloud based restaurant management systems. DineFlow contains a main server and consecutive operational devices that are connected via local network acting in sync for better operation and coordination. A PC can be used as a main server and for other operational devices that are required to fulfill other roles such as waiter or manager, both mobile and tablets can be used effectively. The system uses PostgreSQL as a centralized database for storing and fetching data and Flask SocketIO is used for real time communication and order tracking [4]. Machine learning is also applied onto the historical data for forecasting demand and supplies.

3.3 Architecture layer - user layer

This layer determines the 5 roles that will interact with the DineFlow system mainly, Admin, Manager, Waiter, Cashier, Chief, Kitchen Display System.

Admin will have all the access and necessary information regarding the system, employee, menu, table, order and kitchen, and will act as a centralized administrator to manage and coordinate in times of crises.

Manager is given the access of menu management, table management, order management and report management for effective management execution.

Waiter will have access to table management and order execution.

Cashier will navigate the report and billing section for effective invoicing and billing.

Chief or Kitchen Display System will prepare and update the order preparation status, enabling all the other units to track the order status in real time.

RBAC, Role Based Access Control, is used to ensure that each role or unit is given the access that is necessary to perform the roles that are assigned to them . This will further help reduce unauthorized access to operational and administrative functions[9].

3.4 Local area network layer

This layer is dedicated to setup of local area network that is necessary to eliminate the need of internet connectivity for coordination. The local WiFi is first setup after which Flask is used to run the application on a device which is connected to the same WiFi, generating a shareable LAN address which further can be used by different devices joined from the same WiFi to join and authenticate onto the system with appropriate credentials. The LAN enables application access, order communication and coordination while centralized server maintains the operational data.

3.5 Core modules

Menu management: The complete information about the menu such as menu items, prices and other relevant details will be stored here which can be accessed by the authorised staff members. This module provides centralised access to the menu during operational tasks such as the ordering process, making it easy for the waiter to take and record orders.

Table management: This module stores the information about tables including table number, table status, i.e. whether the table is available or not, and makes it easier for the manager to organise the restaurant, increasing operational capabilities of the restaurant.

Order management module: This module serves the purpose of storing the order information, whether the order is completed or pending, different instructions that the customer has given at the time of giving the order and whether the order is billed or not. Such information can be displayed and updated in this module.

Note: The access to update the orders is reserved for Admin and Manager, and the length of order updates are also determined in advance to avoid malicious and unauthorised activities.

Kitchen display system: The KDS involves the preparation of the order, displaying how long it will take for a chef to complete an order, estimated duration and also deciding the priority of the order. The order can be tracked in real time.

Billing module: This module contains all the billing related tasks such as calculating GST, taxes and discount via percentage or other mediums. The final invoice can be printed and the information can be exported via CSV file.

Employee management: This module gives exclusive access to the Admin to manage employees, update their status, active or inactive, store their important credentials for professional workflow and better management.

Setting module: This module exclusively gives access to change the important details about the restaurant such as name and other credentials like address etc.

Report and analytics module: This module involves the historical data of the restaurant's past transactions and records of past performances of the restaurant in the format of various pictorial representations like charts etc, which can be further used to forecast the demand and revenue of the restaurant.

3.6 Websocket communication layer

The DineFlow system uses Flask SocketIO for real time event driven communication [4].This contributes to better coordination and communication between KDS and order management unit, the order can be tracked and updated in real time, leaving very less room for delays, causing better execution.

3.7 Data layer

The application server is connected to PostgreSQL which is the primary relational database that provides centralised storage, storing operational data and information regarding the system. Application server performs the read and write operations on the database, storing different categories of data independently but in a centralised manner, e.g. user, role, menu items, tables, orders, order items etc. This module ensures that different modules can possibly operate on the same transactional and operational data instead of isolated record maintenance.

4. Machine Learning

Overview

A local ML model is trained on historical transaction records and data of the restaurant to forecast and predict the demand and sales of the restaurant [3] [5]. The forecasting can be of the most sold food item, ingredient that will be demanded the most, or the revenue that the restaurant can potentially generate in the near future. The complete model works in an offline LAN based environment, so there is no need for external API or AI services.

Data collection and preprocessing

The data that is needed for model training is collected from the historical data and records of the restaurant. The data includes information about orders and transactions of the restaurant such as menu items, order ID and frequency, quantity, sales information etc. The data is cleaned on the basis of order execution, meaning cancelled order information and payment data is treated as false and incorrect data and is removed. With that, a low data protection mechanism is also added for security purposes.

Low data security = If item selling days < 2 or total units sold < 3, then item marked as Insufficient data.

Feature engineering

The model heavily relies on 5 features altogether for forecasting and prediction.

1. Day_index: shows the chronological position of the observation.

2. Rolling_3day_avg: calculates the average of short term demand of orders in the last 3 days.
3. Is_weekend: this is a binary feature that determines whether there is a weekend or not that could affect the sales.
4. Hist_daily_avg: demonstrates the average data of historical data on the parameter of daily basis.
5. Recent_7day_avg: shows the average demand of the last 7 days.

Demand forecasting

The Random Forest Regression algorithm from scikit learn library is used to initiate ensembles of decision trees to perform supervised learning on the historical data [6] [7]. There are mainly two reasons why this algorithm is selected.

1. To capture the non linear relations between the historical demand and the food items on the menu.
2. It is both effective and suitable for local execution.

But the model does not completely depend on this algorithm alone. In case of a failure in this mechanism, there is another fallback mechanism introduced that will first calculate the baseline probability on the basis of observation and then a final prediction is calculated on the basis of this baseline prediction.

$$P_{\text{baseline}} = 0.6R_7 + 0.4H \quad (1)$$

R_7 = Recent seven days average.

H = Historical daily average.

$$P_{\text{final}} = 0.7P_{\text{ml}} + 0.3R_7 \quad (2)$$

This mechanism will come into play if the scikit-learn is unavailable.

Model training and evaluation

This section is dedicated to the training method of the model. The traditional training standards are used for efficient training and evaluation. The dataset is divided into two parts, approximately 80 to 20%, where 80% of the data is used for training and the remaining 20% is used as testing data. The implemented evaluation calculates matrix for baseline comparison. This matrix includes Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Coefficient of Determination (R2) after calculating almost 3409 records [8].

The resulting parameters are given below.

Table 2 - Model training and evaluation results

Metric	Value
Records processed	3,409
MAE	0.88 servings
RMSE	1.18 servings
R^2	0.022
Baseline MAE	0.88 servings

Here is what each parameter shows :

Table 3 - Description of evaluation metrics

MAE	Average error
RMSE	Weights of large and significant error
R^2	Whether the underlying pattern is understood by the model or not

Demand prediction and trend analysis

The prediction is done on the basis of two parameters.

1. Next 7 days.
2. Next 30 days.

Each prediction is further stored in prediction table on the basis of three classifications, namely increasing, stable and decreasing, showing the varying demand of the observation.

Predictive inventory intelligence

The predictive intelligence is not just limited to the food item, it extends beyond that and also includes the ingredients required for each food item. This is calculated on the basis of a simple formula.

If (required ingredient) > (available stock), then alert generated for the ingredient.

Predictive formula:

Predictive ingredient requirement = predicted dish demand × ingredient quantity per serving.

Sales and revenue prediction

Revenue forecast is performed after combining the unit price and predicted quantity, along with training done on the historical record that also outlines factors affecting sales like weekend sales or seasonal holiday sales etc.

Formula:

Predicted Revenue = Σ (Predicted Quantity$_i$ × Unit Price$_i$), for i = 1 to n.

5. Functional data flow

Order flow

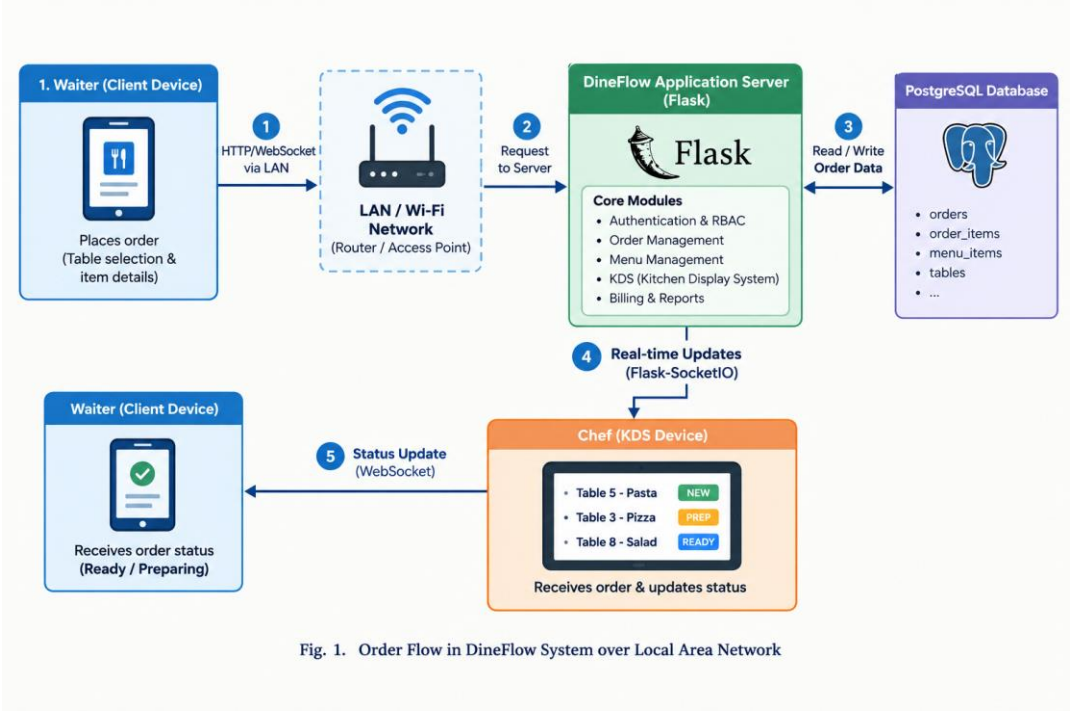


Fig. 1. Order Flow in DineFlow System over Local Area Network

Fig. 2 - Order flow

The order flow exist in order;Waiter → LAN/Wi-Fi → Flask Server → PostgreSQL → Flask-SocketIO → Chef KDS → Order Status Update → Waite

Billing flow

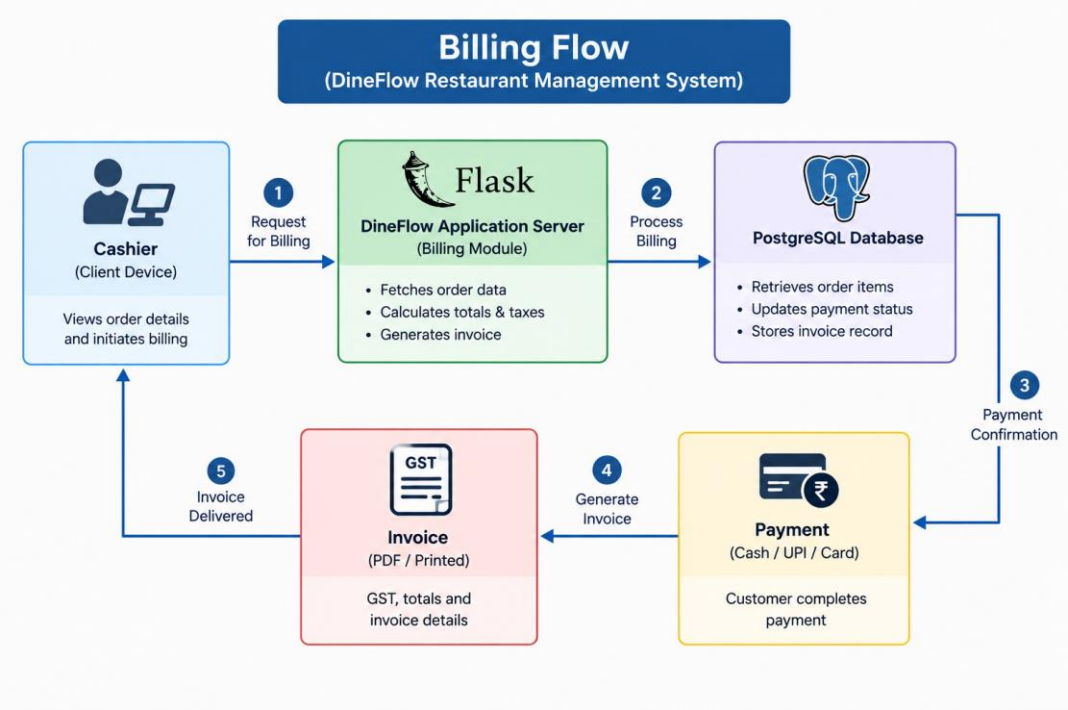


Fig. 3 - Billing flow

The billing flows exist in the order;Cashier → Flask Billing Module → PostgreSQL → Bill & GST Calculation → Payment → Invoice Generation → Invoice Record

Machine learning flow

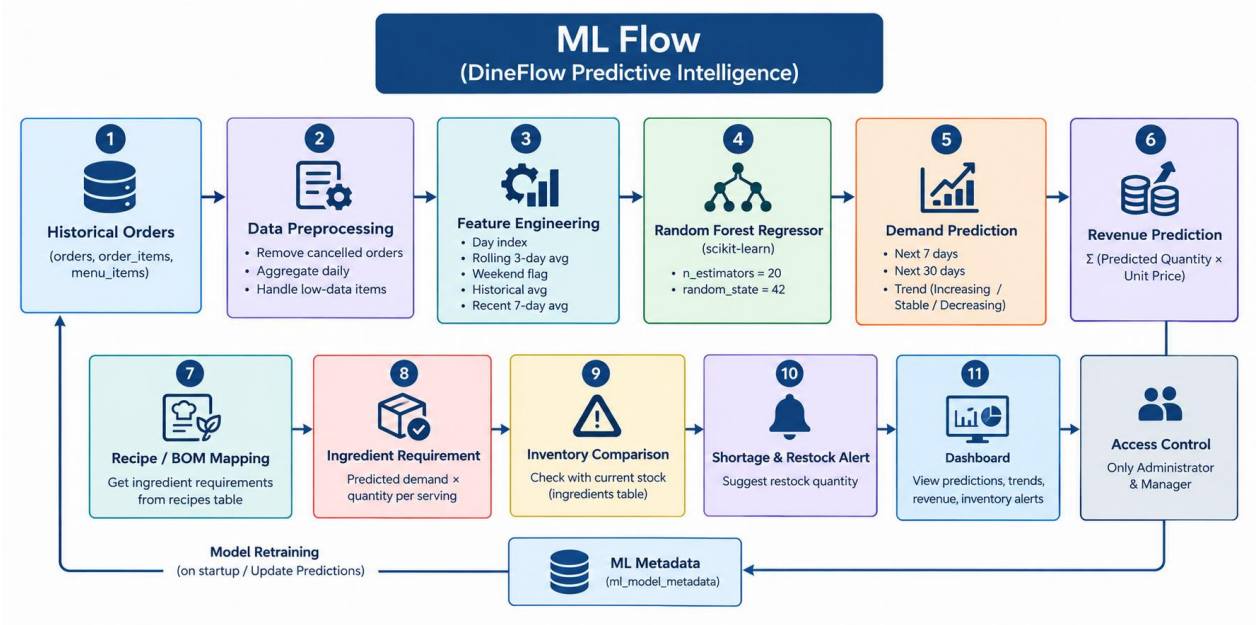


Fig. 4 - ML flow

The machine learning workflow is as follow as :Historical Orders → Data Preprocessing → Feature Engineering → Random Forest Model → Demand Prediction → Revenue Forecast → Recipe/BOM Mapping → Ingredient Requirement → Stock Comparison → Shortage/Restock Alert → Analytics Dashboard

6. Research Methodology

6.1 Research approach

The purpose of this study was to specifically target the problem of internet dependency and dependency on cloud architecture when dealing with restaurant management systems. Therefore, throughout the research, the design of the system had been kept simple and offline first with no interdependency. The implementation of this system was based on Python and Flask, where Python acted as a strong backend and Flask provided connectivity support to achieve real time coordination. The implementation was further tested on specific areas such as KDS, order management, table occupancy management etc. to ensure smooth and efficient performance.

6.2 System development

To fulfill the objective of this study, the architecture was kept local area network based so that maximum network dependency shall be excluded. Flask is selected to establish efficient connectivity between front end, backend and database to achieve maximum stability. PostgreSQL is selected as the primary relational database to optimize the organised data which can be further used for forecasting. Flask SocketIO is used to build real time coordination between several roles and sections, which were effectively implemented because of role based access control to prioritize authorization and security. To evaluate the modules, the data was created and then sustained, which further was used and optimised by the machine learning model for prediction and forecasting.

6.3 Data Processing, ML Development and Evaluation

The dataset that is required for the ML evaluation is taken from the historical records of the restaurant, which is cleaned and processed. The orders that are cancelled are treated as outliers and are removed from the dataset. The necessary metrics such as Mean Absolute Error, Root Mean Squared Error [8] and R squared are calculated, which are required for the forecasting, as this research is not only dedicated to management and organising prior tasks in a traditional restaurant management system in an effective technical way, but also to open scope to indulge modern technology for the growth of business models such as forecasting the revenue by identifying the strengths and weaknesses of the business model.

Then the forecasting is done on the horizon of next week or next month. The same model can be used to optimise the ingredient or inventory management for shortage alerts and prediction towards growing demand and possible required stock of ingredients for food items in the restaurant.

7. Results

7.1 System result

Authentication and role based access control have been successfully implemented in the proposed system, giving appropriate access to different roles like Admin, Manager, Waiter, Chief and Cashier to perform their roles. Each role is required to authenticate using credentials and manage their responsibilities on the dashboard. Fig 5. shows the dashboard dedicated to Admin for managing restaurant operations and orders effectively. The entire system is operational via LAN based connectivity, meaning no cloud dependency. The billing and GST modules are also implemented that efficiently help in calculating taxes and discounts and create printable receipts or bills for completed orders. The connectivity and coordination with KDS is also set up to be real time, enabling near-instantaneous order status updates between placing the order and receiving it on the kitchen side.

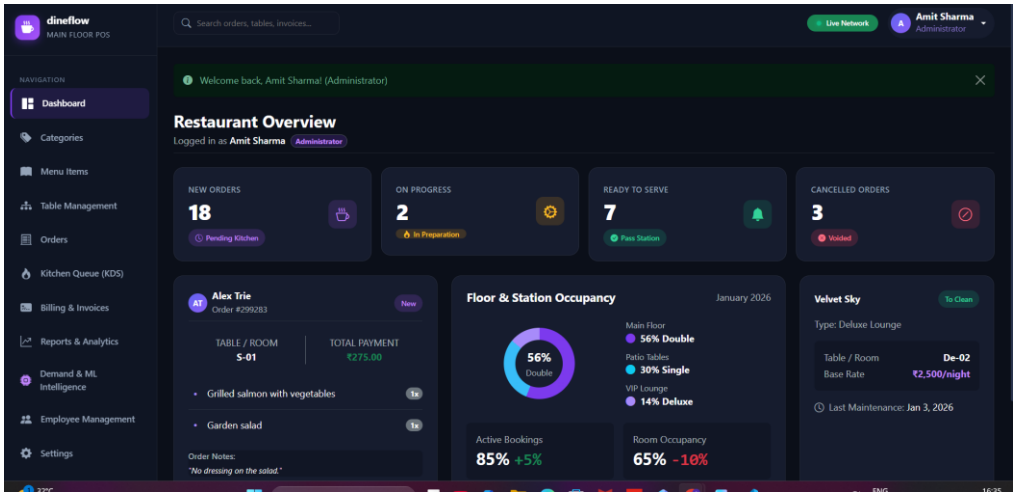


Fig. 5 - DineFlow dashboard

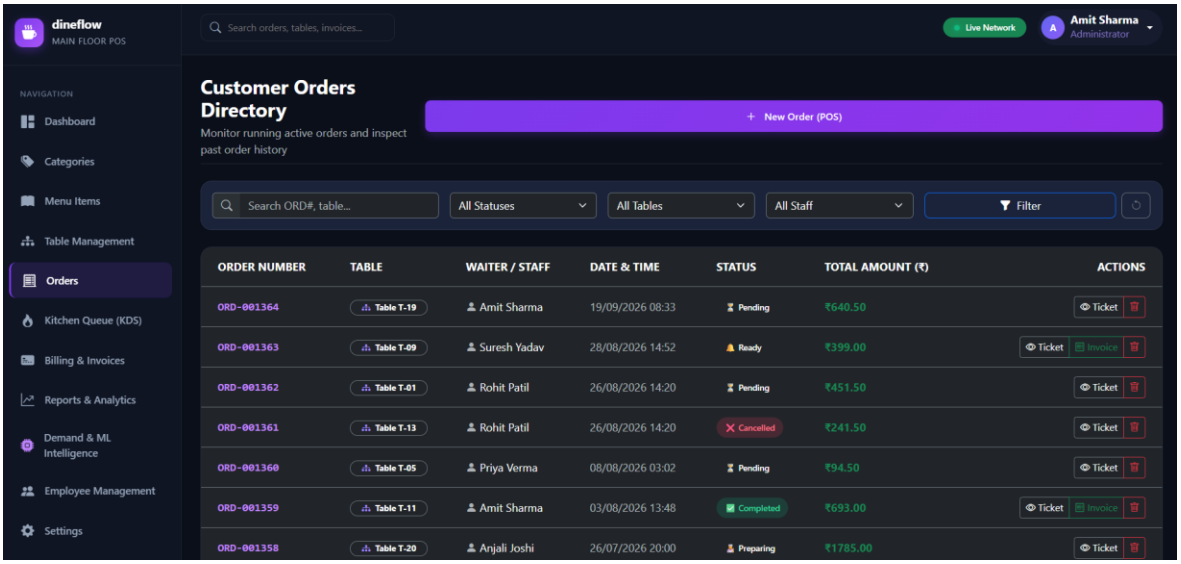


Fig. 6 - DineFlow order management

7.2 ML results

The machine learning model is trained on the prior historical dataset of the restaurant, thus processing 3409 records to derive predictions and forecasting. The dataset is divided into two sections in the ratio of 80:20, where 80% of the data is used for training and 20% of the data is used for testing. The predictions and forecasting are derived on two horizons, namely 7 days and 30 days. The parameters that are used to derive this result include Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and R squared, i.e. Coefficient of Determination (R2).

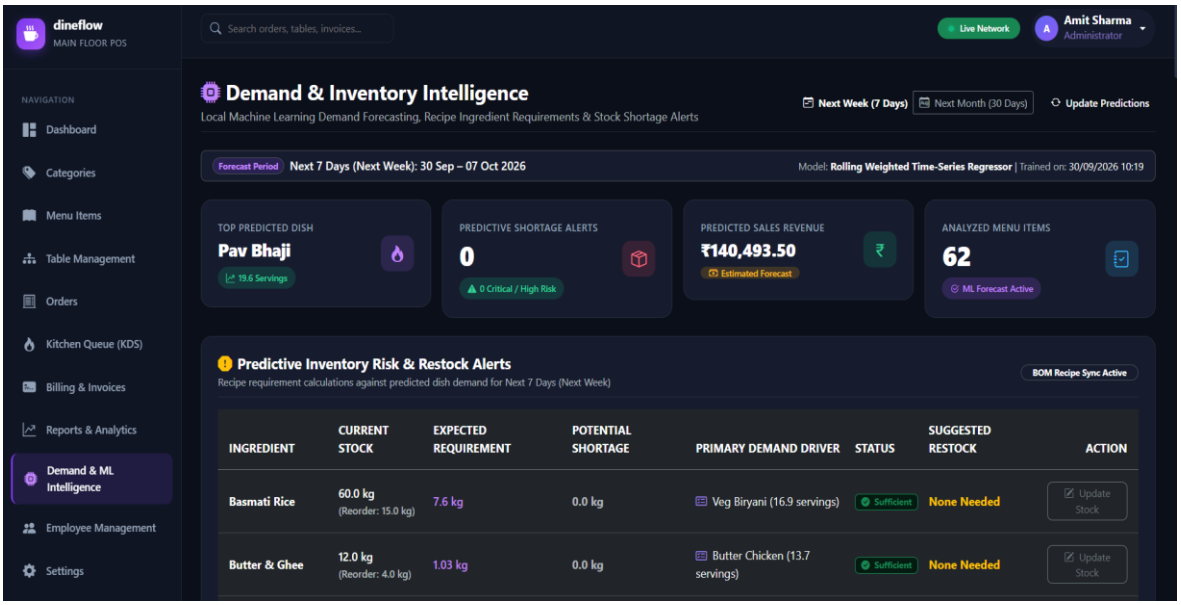


Fig. 7 - ML module

Fig 7 shows predictive output of DineFlow system. The demand forecasting is demonstrated along with latest trends such as increasing, decreasing and stable that are derived from historical transactions. The revenue forecasting, ingredient demand analysis along with shortage alerts are also displayed.

8. Conclusion

This paper concludes the design and successful implementation of DineFlow, an offline-first restaurant management system. This system is completely offline and based on a local area network connectivity and consists of an active role-based access control for secure and authenticated access. The architecture of the system prioritizes keeping operations transparent and organised. The system achieved real-time coordination and communication successfully. The system also optimises machine learning to forecast revenue, sales and demand and it consists of an inventory intelligence to further boost the efficiency and operability of restaurant management.

References

[1] B. Vinodhini, K. Abinaya, R. Roja, and M. Rajeshwari, "Wireless two-way restaurant ordering system via touch screen," *Int. J. Eng. Sci.*, vol. 3, no. 7, pp. 01–05, 2014.

[2] M. Kleppmann, A. Wiggins, P. van Hardenberg, and M. McGranaghan, "Local-first software: You own your data, in spite of the cloud," in *Proc. ACM SIGPLAN Int. Symp. New Ideas, New Paradigms, Reflections Program. Softw. (Onward!)*, Athens, Greece, Oct. 2019, pp. 154–178, doi: 10.1145/3359591.3359737.

[3] T. Tanizaki, T. Hoshino, T. Shimmura, and T. Takenaka, "Restaurants store management based on demand forecasting," *Procedia CIRP*, vol. 88, pp. 580–583, 2020, doi: 10.1016/j.procir.2020.05.101.

[4] I. Fette and A. Melnikov, "The WebSocket Protocol," IETF, RFC 6455, Dec. 2011, doi: 10.17487/RFC6455.

[5] A. Schmidt, M. W. U. Kabir, and T. Hoque, "Machine learning based restaurant sales forecasting," *Mach. Learn. Knowl. Extr.*, vol. 4, no. 1, pp. 105–130, 2022, doi: 10.3390/make4010006.

[6] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[7] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011, doi: 10.48550/arXiv.1201.0490.

[8] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature," *Geosci. Model Dev.*, vol. 7, no. 3, pp. 1247–1250, 2014, doi: 10.5194/gmd-7-1247-2014.

[9] D. F. Ferraiolo and D. R. Kuhn, "Role-based access controls," in *Proc. 15th Nat. Comput. Security Conf.*, 1992, pp. 554–563.