



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

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

^aDepartment of Computer Science and Engineering, Prof. Ram Meghe Institute of Technology & Research

ABSTRACT :

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: Restaurant management system, Offline-first, Local area network, Flask, PostgreSQL, Random Forest, Demand forecasting, Role-based access control

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 (Vinodhini et al., 2014). 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 (Kleppmann et al., 2019). 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.

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 (Kleppmann et al., 2019).

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 (Tanizaki et al., 2020).

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.

Literature review

Vinodhini et al. (2014) 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. Kleppmann et al. (2019) 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. Tanizaki et al. (2020) introduces the idea of using POS data from restaurants to generate demand forecasts using algorithms like Random Forest. However, real time communication is still a concept that needs to be applied for effective order execution. Fette and Melnikov (2011) provided an appropriate solution by introducing the idea of WebSocket, which directly contributes to real-time communication between different sections of the system. Schmidt et al. (2022) 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. Breiman (2001) explained Random Forest in detail, describing fundamental approaches to effectively implementing ensemble learning and combining multiple decision trees to derive accurate and realistic predictions. Pedregosa et al. (2011) 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. Chai and Draxler (2014) discussed the importance of MAE and RMSE and argued that both should be used together to optimize the evaluation. Ferraiolo and Kuhn (1992) 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.

Table 1 - Summary of the reviewed literature

Work	Main focus	Observed limitation	Implication
Vinodhini et al. (2014)	Wireless touch-screen restaurant ordering	Not offline/LAN-based; internet dependency not addressed	Supports need for a fully offline coordination layer
Kleppmann et al. (2019)	Local-first software design principles	Not restaurant-specific; generic software architecture	Motivates offline, LAN-based system design
Tanizaki et al. (2020)	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 (2011)	WebSocket protocol for real-time communication	Generic protocol, not restaurant-domain specific	Provides technical basis for real-time KDS updates
Schmidt et al. (2022)	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 (2001)	Random Forest algorithm (foundational)	Not domain-specific; general-purpose ensemble method	Provides theoretical basis for chosen ML algorithm
Pedregosa et al. (2011)	Scikit-learn ML library	Generic toolkit, not tailored to restaurant use case	Enables local, offline execution of ML model
Chai and Draxler (2014)	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 (1992)	Role-Based Access Control (foundational)	Generic model, not restaurant-role specific	Supports RBAC implementation for Admin/Manager/Waiter/Cashier/Chef roles

Proposed system architecture and functional modules

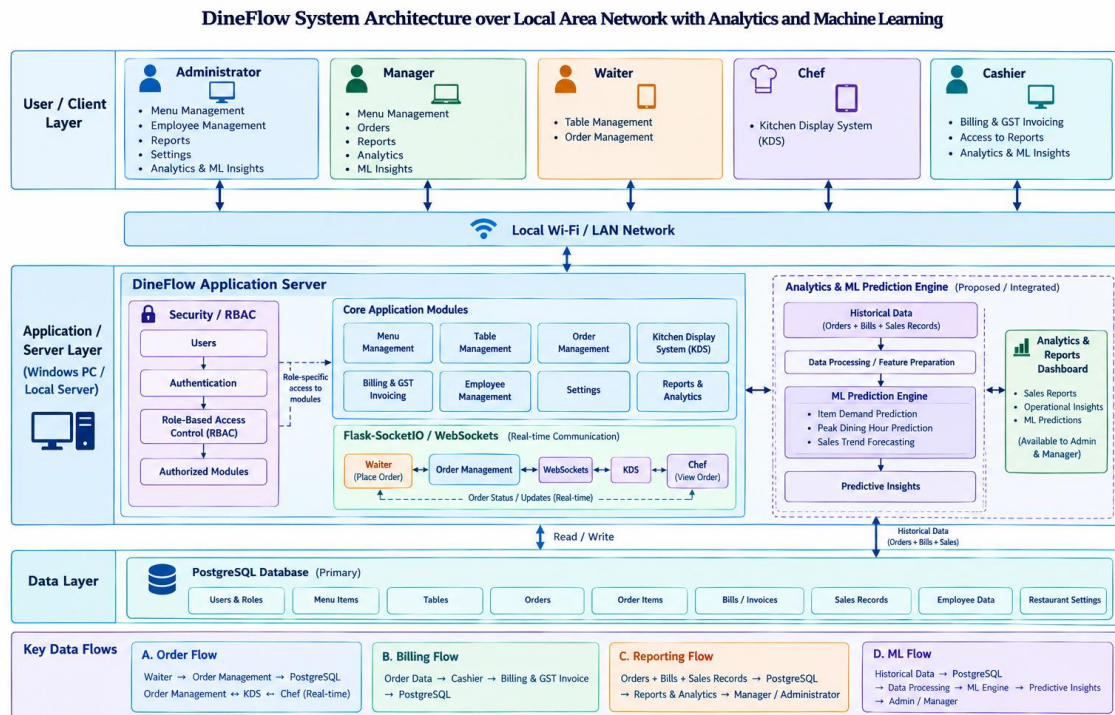


Fig. 1 - DineFlow System Architecture over Local Area Network with Analytics and Machine Learning

1.1. 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 (Fette & Melnikov, 2011). Machine learning is also applied onto the historical data for forecasting demand and supplies.

1.2. 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 (Ferraiolo & Kuhn, 1992).

1.3. 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.

1.4. 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.

1.5. WebSocket communication layer

The DineFlow system uses Flask SocketIO for real time event driven communication (Fette & Melnikov, 2011). 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.

1.6. 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.

1.7. Machine learning

1.7.1. 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 (Tanizaki et al., 2020; Schmidt et al., 2022). 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.

1.7.2. 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.

1.7.3. 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.

1.7.4. 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 (Breiman, 2001; Pedregosa et al., 2011). 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.

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

R_7 = Recent seven days average.

H = Historical daily average.

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

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

1.7.5. 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) on the collected records (Chai & Draxler, 2014).

The parameters that will be recorded are given below.

Table 2 - Metrics to be recorded for the demand forecasting model

Metric	Value
Records processed	Final dataset size, to be counted
MAE	To be measured (servings)
RMSE	To be measured (servings)
R ²	To be measured
Baseline MAE	To be measured (servings)

Here is what each parameter shows :

Table 3 - What each parameter shows

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

1.7.6. 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.

1.7.7. 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.

1.7.8. 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 = $\sum (\text{Predicted Quantity}_i \times \text{Unit Price}_i)$, for $i = 1$ to n .

1.8. Functional data flow

1.8.1. Order flow

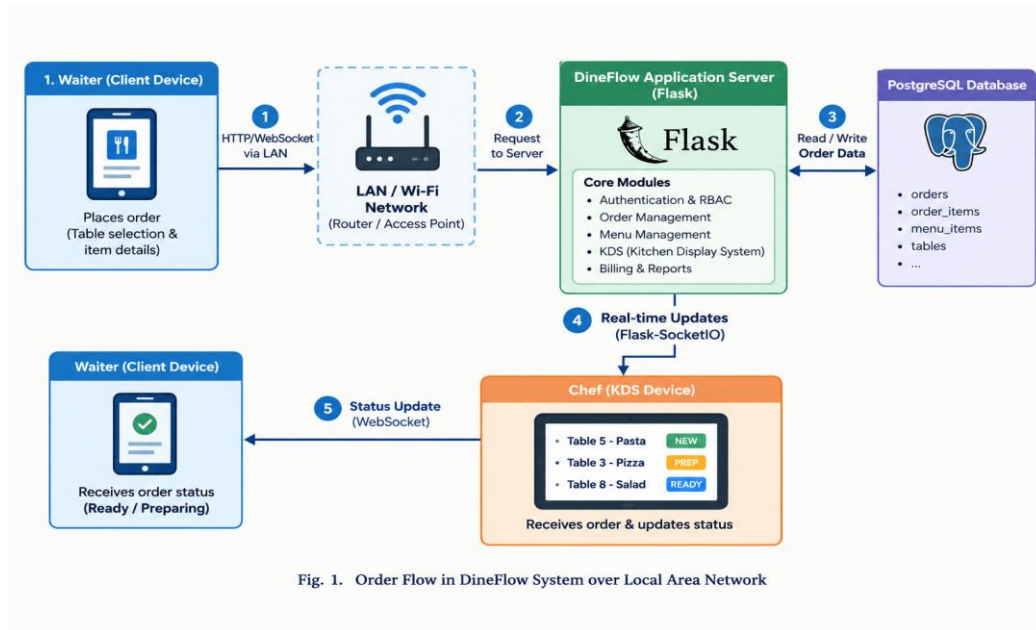
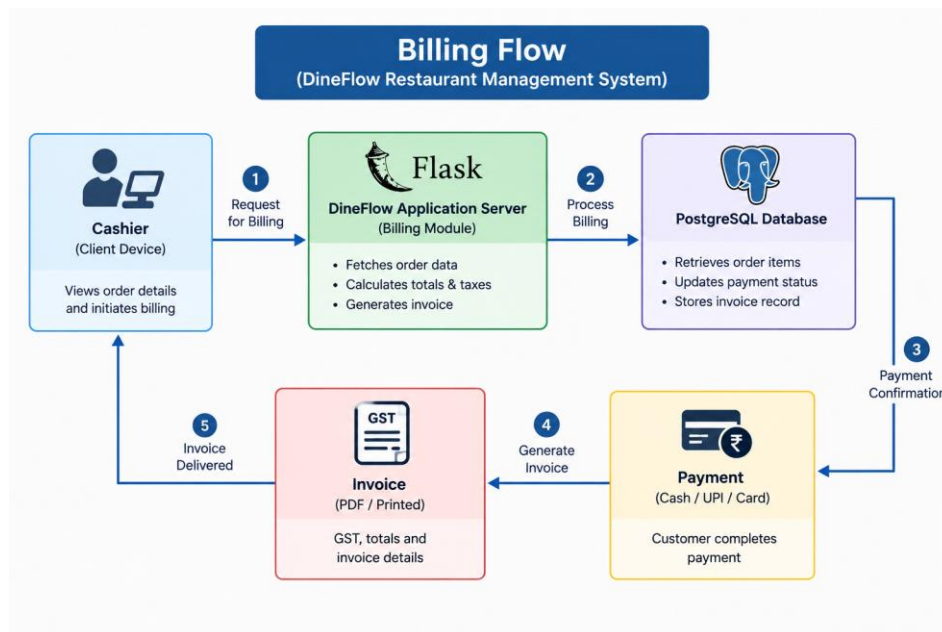


Fig. 2 - Order flow in DineFlow over the local area network

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

1.8.2. Billing flow



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

1.8.3. Machine learning flow

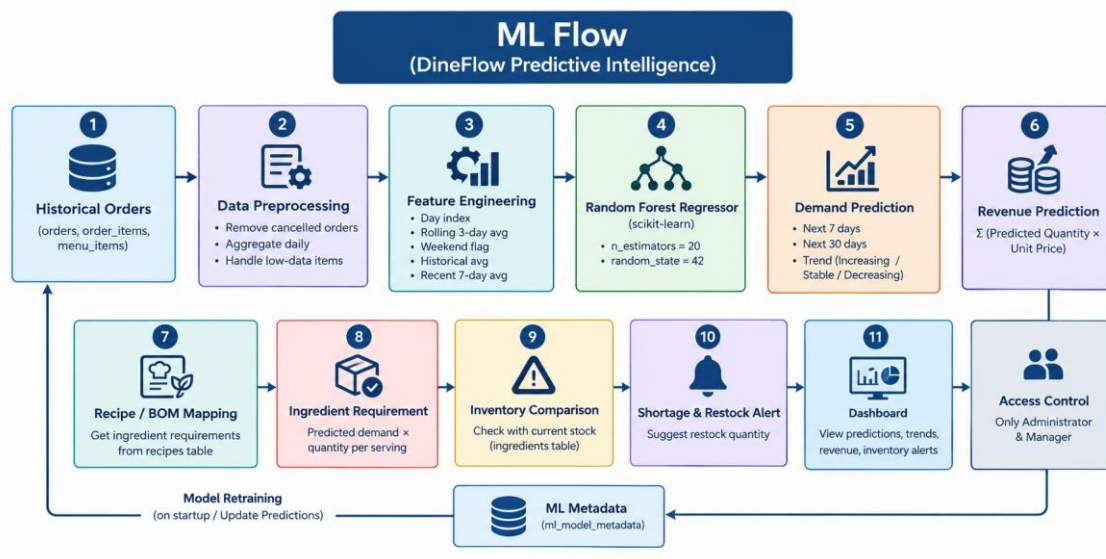


Fig. 4 - Machine learning flow in DineFlow

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

Research methodology

1.9. 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 will be tested on specific areas such as KDS, order management, table occupancy management etc. to ensure smooth and efficient performance.

1.10. 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.

1.11. 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 (Chai & Draxler, 2014) 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.

Expected results

1.12. Expected system results

At this stage DineFlow is a proposed system, so the results discussed here are the outcomes we expect once all the modules are built and tested on a local network. The first thing to be checked is authentication and role based access control. Each of the five roles, Admin, Manager, Waiter, Cashier and Chef, should be able to log in with its own credentials and reach only the screens that belong to its work. For example, a waiter should not be able to open the employee or settings pages.

The whole system is expected to run on one local WiFi network, with the application server on a single PC and the other devices joined through the shared LAN address. To confirm the offline claim, the plan is to cut the internet connection of the router and run a full day of dummy orders, starting from table selection and ending at billing. If the design is correct nothing should change, because no module depends on a cloud service.

For billing, the GST and discount totals are expected to match the same bills calculated by hand, and a printable invoice along with a CSV export should be generated for every completed order. The KDS connection through Flask SocketIO should show a new order on the chef screen almost immediately after the waiter submits it, and the status changed by the chef should reach the waiter in the same way. The time between these events will be noted on test orders, so that the delay can be reported as a number and not only as a general statement. Table 4 lists what will be checked for each module.

Table 4 - Planned checks for the system modules

Module	What will be tested	Expected outcome
Role based access	Login and page access for each of the five roles	Every role reaches only its permitted pages
LAN operation	A complete order cycle with the internet disconnected	No loss of function without internet
Billing	GST and discount totals compared with manual calculation	Totals match; invoice and CSV are generated
Kitchen display system	Delay between order placement and the chef screen update	Updates appear in near real time
Inventory alerts	Required ingredient quantity compared with stock in the database	Alert is raised when stock is short

1.13. Expected ML results

The demand model will be trained on the historical transaction records collected for the restaurant, using the 80:20 split of training and testing data described earlier. Forecasts will be produced for the 7 day and 30 day horizons, and every item will be marked as increasing, stable or decreasing in the prediction table. MAE, RMSE and R^2 will be calculated on the testing part of the data.

Item level restaurant demand is noisy, and Schmidt et al. (2022) already observed that simple models can come close to complex ones when the dataset is small. So we do not assume that Random Forest will win by a wide margin. The comparison that matters is the one against the baseline prediction built from the recent 7 day and the historical averages, and the model is expected to be at least as good as that baseline on MAE and RMSE. A low R^2 on the first dataset would not be a surprise either, since a few months of sales data cannot explain much of the day to day variation. If that happens it will be reported as it is, because MAE, RMSE and R^2 are meant to be read together (Chai & Draxler, 2014).

On the analytics dashboard we expect the demand forecast with its trend labels, the revenue forecast, the ingredient requirement for the coming days and the shortage alerts to be available in one place. The predicted ingredient requirement will be compared with the stock entered in the system, and the alerts raised will be checked against a few cases worked out manually. Once the system is running with the restaurant's own data, these expected results will be replaced with the measured ones.

Conclusion

This paper presents the proposed design of DineFlow, an offline-first restaurant management system. The system is planned to be completely offline and based on local area network connectivity, with an active role-based access control for secure and authenticated access. The architecture of the system prioritizes keeping operations transparent and organised. The system is expected to achieve real-time coordination and communication between the different sections of the restaurant. It also applies machine learning to forecast revenue, sales and demand, and includes an inventory intelligence to further boost the efficiency and operability of restaurant management. Testing with real restaurant data is the next step of this work.

REFERENCES

- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. <https://doi.org/10.1023/A:1010933404324>
- Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3), 1247–1250. <https://doi.org/10.5194/gmd-7-1247-2014>
- Ferraiolo, D. F., & Kuhn, D. R. (1992). Role-based access controls. In *Proceedings of the 15th National Computer Security Conference* (pp. 554–563).
- Fette, I., & Melnikov, A. (2011). *The WebSocket Protocol* (RFC 6455). Internet Engineering Task Force. <https://doi.org/10.17487/RFC6455>

-
- Kleppmann, M., Wiggins, A., van Hardenberg, P., & McGranaghan, M. (2019). Local-first software: You own your data, in spite of the cloud. In *Proceedings of the ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!)* (pp. 154–178). Athens, Greece. <https://doi.org/10.1145/3359591.3359737>
 - Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830. <https://doi.org/10.48550/arXiv.1201.0490>
 - Schmidt, A., Kabir, M. W. U., & Hoque, T. (2022). Machine learning based restaurant sales forecasting. *Machine Learning and Knowledge Extraction*, 4(1), 105–130. <https://doi.org/10.3390/make4010006>
 - Tanizaki, T., Hoshino, T., Shimmura, T., & Takenaka, T. (2020). Restaurants store management based on demand forecasting. *Procedia CIRP*, 88, 580–583. <https://doi.org/10.1016/j.procir.2020.05.101>
 - Vinodhini, B., Abinaya, K., Roja, R., & Rajeshwari, M. (2014). Wireless two-way restaurant ordering system via touch screen. *International Journal of Engineering Sciences*, 3(7), 1–5.