UMHACKTHON 2025

Domain 3: Economic Empowerment through AI
Re-engage, Reconnect, Rebuild: Smarter Sales with AI

Group Name: Nth times the charm

Group Members: Kaviraj Vijayanthiran

Angel Tan Xian Theng

Lim Shen Yik

Siew Jun Zhen

Table Of Contents

🏷️ **Re-engage, Reconnect, Rebuild: Smarter Sales with AI**

In the fast-paced world of food and beverage services, merchants face significant challenges in maximising sales while ensuring customer retention. Our project combines two innovative solutions to tackle these challenges with the power of AI. Introducing **GrabBack,** a Smart Re-Engagement Promo Generator, and **GrabCast,** an AI-Powered Daily Sales Forecast.

1.0 Idea Overview

Re-engage, Reconnect, Rebuild: Smarter Sales with AI" introduces two AI-powered solutions designed to benefit all Grab merchants (MSME & Large Merchant Chains) — small businesses with limited resources and large corporations with bulk operations. **GrabBack** is the first solution, focused on re-engaging lapsed customers by delivering hyper-personalised promotional offers, driving customer retention, and increasing sales with minimal effort. This is especially valuable for small merchants who lack the resources for complex marketing strategies. **GrabCast** uses AI to provide accurate, data-driven daily sales forecasting. For small merchants, this helps manage inventory efficiently, while large corporations can optimise production levels and reduce food waste by accurately predicting demand. Both solutions work together to streamline operations, enhance customer relationships, and ultimately drive smarter sales.

GrabBack: ***Customers left? Let's grab them again.***
GrabBack is a Smart Re-Engagement Promo Generator that helps merchants win back lapsed customers by generating personalised promotional offers based on order history. With AI-driven insights, the tool identifies customers who haven't purchased recently, suggests their favorite item, and creates targeted offers to bring them back. Merchants can easily re-engage customers through automated promotions, boosting loyalty and increasing sales.

GrabCast: ***Unlock the power of tomorrow's sales today.***
GrabCast is an AI-Powered Daily Sales Forecasting tool that uses historical sales data from the Grab platform to help merchants forecast daily sales. It recommends optimal inventory levels, preventing overstocking and stockouts, ensuring smarter decision-making and streamlined operations. This solution empowers merchants to better prepare for each day's sales without needing technical skills.

By combining GrabBack and GrabCast, this comprehensive AI-powered solution helps merchants plan smarter, reduce waste, increase revenue, and build lasting customer relationships - empowering all businesses within the Grab ecosystem with user-friendly, accessible technology.

2.0 Unique Selling Points (USP)

What makes this idea new, different, or a breakthrough:
Introducing an AI-powered solution that transforms how Grab merchants manage their **customer engagement and inventory planning**.

- **GrabBack** uses customer transaction_data & transaction_item & item datasets to create personalized, targeted promotions that win back lapsed customers automatically.

- **GrabCast** leverages AI to forecast daily sales at the item level, enabling merchants to optimize food prep and inventory, reducing waste and maximizing profit.

By integrating both of these into one intuitive platform within the Grab ecosystem, we empower small merchants with powerful tools previously only accessible to large chains.

Creative Twist or Clever Feature:
The true innovation lies in the **synergy** between **customer re-engagement and AI-driven sales forecasting** — a first in this domain.

- **GrabBack** generates hyper-targeted offers based on individual buying behavior.

- **GrabCast** predicts demand in advance, so merchants can prepare the right amount of food at the right time.

Together, this means smarter operations, fewer losses, and stronger customer loyalty — all with just a few taps.

Why It Hasn't Been Done Yet / Why Now Is the Right Time:
While AI tools for forecasting and CRM exist, they are often expensive, complex, and built for enterprise users. With more MSMEs embracing digital platforms like Grab, now is the time to provide **affordable, accessible AI tools** that simplify decision-making for time-strapped, non-technical merchants.

Why Grab is First:

- Grab is uniquely positioned to lead this transformation by integrating **GrabBack** and **GrabCast** directly into the **existing merchant app**, using rich in-platform data.

- No third-party logins. No extra cost. Just intelligent automation that helps small businesses grow, save, and thrive — right from the app they already use.

3.0 Competitor Analysis

Feature/Functionality	Grab	Foodpanda	Gojek	Swiggy
AI-Powered Sales Forecasting	GrabCast: Predicts daily sales, optimizes inventory based on historical data.	No advanced forecasting; offers basic sales insights.	No forecasting tool for merchants.	Basic sales data, no AI-driven forecasting.
AI-Driven Customer Re-Engagement	GrabBack: Creates hyper-targeted, personalized offers based on order history.	Offers discounts, but no personalized AI-driven engagement.	No AI-powered customer re-engagement tools.	Basic promotional tools, no AI-driven targeting.
Integration with App Ecosystem	Fully integrated, user-friendly for small merchants in Grab's ecosystem.	Standalone app, no deep integration with other business tools.	Operates as a multi-service app, lacks AI-powered merchant tools.	Focuses on food delivery; no integrated merchant tools.
Target Audience	All Grab merchants are looking for AI-driven tools for forecasting	Merchants on Foodpanda, but lacks personalized AI tools.	Multi-service app, mainly focused on larger businesses.	Primarily focuses on larger merchants in food delivery.

	and engagement.			

4.0 Problem, Solution, and Impact

Business Goals: Increase customer retention and sales through personalized engagement, optimize inventory to reduce waste, and drive operational efficiency through AI-powered solutions.

Proposed Solution	**GrabBack**	**GrabCast**
Real-World Problem	Many merchants are unaware when customers drop off and have no tools to bring them back. Ineffective re-engagement and generic discounts lead to lost revenue (Deskera, 2022).	17,000 tonnes of food waste are generated daily in Malaysia, with 4,000+ tonnes still edible. Many small merchants over/under prepare due to lack of data tools (The Star, 2022)
Problem Statement	No CRM or automated re-engagement tool for MSMEs. Leads to lost revenue and customer churn.	Merchants rely on gut feeling for daily prep, resulting in waste or missed sales.
Objectives	Win back lapsed customers using smart, personalised, no-hassle promotions.	Improve inventory prep accuracy through simple AI forecasting.
How the Tool Solves It	Detects lapsed users, identifies past favorites, auto-generates personalized promo messages. Send with one click.	Uses historical data to predict daily demand and recommend optimal prep quantities. Easy to use.
Impact / Value Created	Recovers lost sales. Increases customer loyalty. Requires no marketing skills. Saves time.	Reduces waste. Improves prep accuracy. Boosts operational efficiency. Supports sustainable growth.

Aligned SDG(s)		
How Our AI Tools Align to SDGs	**SDG 9 – Industry, Innovation & Infrastructure** • Introduces an AI-powered tool that replaces manual customer tracking and marketing. • Provides accessible, plug-and-play re-engagement solutions for merchants with no CRM or digital marketing background. • Encourages inclusive digital transformation for small, underserved businesses. **SDG 8 – Decent Work & Economic Growth** • Helps merchants increase customer retention and repeat sales with minimal effort. • Drives higher sales consistency and business sustainability. • Reduces reliance on	**SDG 12 – Responsible Consumption & Production** • Helps merchants reduce food waste through data-driven daily prep forecasting. • Prevents overproduction (which leads to waste) and underproduction (which leads to missed sales). • Encourages smarter, more sustainable inventory planning practices. **SDG 9 – Industry, Innovation & Infrastructure** • Delivers AI-powered forecasting that's easy to use — no analytics or technical skills required. • Modernizes small food businesses by replacing "gut feeling" with intelligent decision-making. • Promotes scalable innovation in everyday merchant operations.

	costly third-party marketing tools or external agencies.	

In-depth Explanation

In Malaysia and across Southeast Asia, many micro, small, and medium-sized large food merchants face operational inefficiencies due to a lack of data-driven tools. A common pain point is **demand forecasting**, many vendors estimate their daily food preparation quantities purely by instinct or past “gut feel”, often leading to **over-preparation**, resulting in food waste, or **under-preparation**, causing missed sales and customer dissatisfaction. In Malaysia alone, over **17,000 tonnes of food waste are generated daily**, with nearly **4,000 tonnes still edible**, much of it coming from commercial food operators including small-scale merchants (The Star, 2022). This inefficiency not only affects profit margins but also contributes to the country’s rising food waste problem.

At the same time, these merchants often struggle with **customer re-engagement and retention**. Many are unaware of which customers have stopped ordering and lack the tools to win them back. According to Deskera (2022), customer churn, ineffective marketing, and limited loyalty strategies are among the top reasons why restaurants fail. Unfortunately, most MSMEs lack access to CRM systems or marketing automation tools, leading to high customer drop-off and lost revenue opportunities. Furthermore, McKinsey’s personalization research highlights that **tailored messaging is 2–3x more effective** at customer conversion compared to generic discounts (McKinsey, 2021), which most MSMEs currently rely on.

Our proposed solution tackles both challenges through two focused AI tools: **GrabCast** and **GrabBack**. **GrabCast** functions as a smart daily sales forecasting assistant, analysing historical order data to recommend min–max preparation ranges for each menu item. This helps merchants **reduce food waste, avoid understocking, and make confident daily planning decisions without requiring any technical expertise**.

On the other hand, **GrabBack** acts as a smart re-engagement engine. It identifies lapsed customers and automatically generates **personalized promotional offers** based on their past purchasing behavior, making it easy for merchants to re-establish

customer relationships and recover lost revenue. Together, these two tools **empower small food businesses** to **run leaner operations and build stronger customer loyalty**.

This initiative strongly aligns with several **United Nations Sustainable Development Goals (SDGs)**. Most directly, it supports **SDG 9: Industry, Innovation and Infrastructure**, by promoting inclusive, sustainable industrialization through accessible, scalable, and easy-to-use AI tools tailored for under-resourced merchants. **GrabCast** and **GrabBack** help bridge the digital divide by enabling micro and small businesses within the Grab ecosystem to make informed, data-driven decisions without technical barriers.

In addition, the solution contributes to **SDG 12: Responsible Consumption and Production**, by reducing food waste through accurate daily prep forecasting, helping merchants optimize inventory and avoid overproduction. It also supports **SDG 8: Decent Work and Economic Growth**, by improving revenue stability and business sustainability for MSMEs, empowering them to grow with minimal risk and overhead.

Together, these solutions enable small merchants to make smarter, data-driven decisions every day, whether it's reducing food waste through better inventory forecasting with **GrabCast**, or increasing customer loyalty through personalized retention strategies with **GrabBack**. By transforming daily operations and customer engagement into guided, AI-assisted workflows, this dual-solution approach empowers small businesses to grow sustainably and thrive in a competitive digital economy.

5.0 Target Users

GrabBack and GrabCast are designed for:

Micro, Small, and Medium Enterprises (MSMEs) operating on the Grab platform, especially in the **F&B industry** who:

- Lack access to CRM systems, marketing tools, or data analytics platforms.
- Struggle with customer retention and repeat orders.
- Rely on instinct or manual tracking for inventory planning.

New Grab merchants who want a simple, plug-and-play tool to get started with smarter operations.

Larger merchant chains using Grab's platform who want to optimize their production levels, reduce food waste, and enhance customer engagement at scale.

Referenced from:

Supported by Deskera (2022), which highlights churn and poor marketing as top reasons small restaurants fail, and The Star (2022), which underscores the food waste crisis in Malaysia due to poor prep planning.

6.0 Tech Stack (Technical Execution)

6.1 Dataset Introduction & Exploratory Analysis

To support our dual objectives, **forecasting daily order value** (GrabCast) and **re-engaging lapsed customers** (GrabBack), we worked with a robust synthetic dataset modeled on real-world food delivery data. This dataset emulates merchant, item, transaction, and customer behaviors within a Grab-like ecosystem.

The project utilizes **two distinct dataset views** tailored for different analytical goals:

- For **GrabCast**, the focus is on item-level, time-based sales patterns to enable forecasting and inventory planning.

- For **GrabBack**, the focus shifts to customer behavior and preferences to enable targeted re-engagement.

To achieve its intended impact, GrabCast is structured around two primary goals focused on forecasting and optimization.

Objective 1: Daily Sales Forecasting Revenue (ARIMA)

1. **Setup & Data Loading:**

- Import necessary libraries (Polars, Skimpy, Seaborn, Matplotlib, Statsmodels, Pandas, Numpy).
- Define connection details for a PostgreSQL database.
- Specify a `merchant_id`.
- Construct an SQL query to select all data for the specified merchant from a view named `combined_order_view`.
- Read the data from the database into a Polars DataFrame (`df`) using `pl.read_database_uri`.

2. **Initial Exploration & Feature Engineering:**

- Clean DataFrame column names using `skimpy.clean_columns`.
- Generate a summary statistics report of the data using `skimpy.skim`.
- Derive new time-based features from the `order_time` column: `order_date`, `order_hour`, `order_weekday`.
- Calculate duration features: `time_to_arrive`, `wait_to_pickup`, `delivery_duration` by subtracting relevant timestamp columns.

3. **Aggregation:**

- Group the main DataFrame (`df`) by `order_date`.
- Calculate daily aggregates: `total_orders` (count of unique order IDs), `total_revenue` (sum of subtotals), and `total_items` (sum of quantities).
- Store these aggregates in a new DataFrame `daily_sales_df` and sort by date.

4. **Data Exploration & Preprocessing (on Daily Data):**

- **Visualization:** Convert `daily_sales_df` to Pandas and create boxplots and histograms for `total_orders`, `total_revenue`, and `total_items` to understand their distributions.

- **Outlier Detection:** Calculate the number of outliers for the numeric columns in `daily_sales_df` using the Interquartile Range (IQR) method.
- **Transformation:**
    - Apply Winsorization (clipping values below the 3rd percentile and above the 97th percentile) to `total_orders`, `total_revenue`, `total_items`.
    - Apply a log transformation (`log1p`) to the Winsorized columns to potentially normalize their distributions.
    - Visualize the distributions of the Winsorized and log-transformed data.
- **Normalization:** Apply Min-Max scaling to the log-transformed columns, scaling them to a range of [0, 1]. Visualize the distributions of the normalized data.

5. **Time Series Analysis & Forecasting (on Daily Revenue):**

- **Visualization:** Plot the original `total_revenue` time series.
- **Stationarity Check:** Plot rolling mean/standard deviation and perform the Augmented Dickey-Fuller (ADF) test on `total_revenue` to check for stationarity.
- **Differencing:** Apply first-order differencing to `total_revenue` to make the series stationary (if needed). Check stationarity again using plots and ADF test on the differenced series.
- **Order Identification:** Plot the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) of the (potentially differenced) series to help determine appropriate `p` and `q` orders for an ARIMA model.
- **ARIMA Modeling:**
    - Split the `total_revenue` series into training and testing sets.
    - Perform a grid search (looping through p, d, q values) to find the best ARIMA model order based on minimizing Mean Squared Error (MSE) or AIC on the test set predictions. (Note: `d` is fixed at 1 based on earlier differencing).
    - Fit the best ARIMA model identified by the grid search on the training data.
    - Generate forecasts on the test set period.
    - Plot the training data, actual test data, and the ARIMA forecasts.
    - Perform residual diagnostics (plotting residuals, histogram, ACF, Ljung-Box test) on the fitted model to check if the model assumptions hold (residuals resemble white noise).
- **SARIMAX Modeling (Seasonal ARIMA with Exogenous Features):**
    - Prepare data by adding potential exogenous variables (`day_of_week`, `promo_flag`) to the daily revenue data.

- Split data into training and testing sets (including exogenous features).
- Fit a SARIMAX model, specifying both non-seasonal (`order` from best ARIMA) and seasonal (`seasonal_order`, e.g., weekly seasonality s=7) components, using the exogenous features.
- Generate forecasts on the test set, providing the corresponding future exogenous features.
- Compare the SARIMAX forecasts with the actual test data, calculate the percentage deviation, and categorize the forecast accuracy ("As Expected", "Better than Expected", "Worse than Expected").
- Display a summary table of the forecast comparison.

Objective 2: Item Quantity Prediction (using XGBoost)

1. **Data Loading and Initial Filtering:**

   - Imports necessary libraries: `polars` for efficient data manipulation, `skimpy` for data cleaning and summary.
   - Defines a connection URI to a PostgreSQL database.
   - Specifies a `merchant_id` to filter data.
   - Constructs a SQL query to select all columns from a `combined_order_view` for the specified `merchant_id`.
   - Uses `pl.read_database_uri` to read the data directly into a Polars DataFrame.

2. **Initial Data Exploration and Cleaning (Optional):**

   - `clean_columns(df)`: Cleans the column names of the DataFrame (e.g., removes special characters, converts to lowercase).
   - `skim(df)`: Provides a summary of the DataFrame's structure and data types.
   - `print(df.head())`: Displays the first few rows of the DataFrame.

3. **Feature Engineering - Date Feature:**

- Creates a new column `order_date` by extracting the date part from the `order_time` column.

4. **Data Aggregation:**

   - Groups the DataFrame by `order_date` and `item_id`.
   - Calculates the `daily_quantity_sold` by summing the `quantity` for each group.
   - Optionally keeps the first `item_name` for each group.
   - Sorts the resulting `daily_item_sales` DataFrame by `order_date` and `item_id`.
   - Prints the head of the aggregated DataFrame.

5. **Feature Engineering - Time-Based Features:**

   - Adds several time-based features derived from the `order_date` column:
     - `day_of_week`
     - `day_of_year`
     - `month`
     - `year`
     - `is_weekend` (binary indicator)
   - Prints the head of the DataFrame with these new features.

6. **Data Visualization (Exploratory):**

   - Converts the Polars DataFrame to a Pandas DataFrame (`pdf`).
   - Uses `seaborn` and `matplotlib` to create:
     - A boxplot of the `daily_quantity_sold` to visualize its distribution and potential outliers.
     - A histogram with a Kernel Density Estimate (KDE) to further understand the distribution of `daily_quantity_sold`.
   - Displays the plots.

7. **Feature Engineering - Lag Features:**

   - Sorts the DataFrame by `item_id` and `order_date`. This is crucial for correct lag calculation within each item's time series.
   - Creates lag features:

    - `lag_1`: The `daily_quantity_sold` from the previous day (shifted by 1).
    - `lag_7`: The `daily_quantity_sold` from seven days prior (shifted by 7).
- The `.over("item_id")` ensures that the shifting is performed independently for each unique `item_id`.
- Prints the head of the DataFrame with lag features.

8. **Data Cleaning - Handling Missing Lag Values:**

- Removes rows where `lag_1` or `lag_7` are null. These null values occur for the first few days of each item's sales history because there are no prior days to lag from.
- Prints the head of the cleaned DataFrame.

9. **Train-Test Split:**

- Converts the Polars DataFrame back to a Pandas DataFrame.
- Sorts the DataFrame by `item_id` and `order_date`.
- Defines a `split_day` as 30 days before the maximum `order_date`.
- Splits the data into `train_df` (data before `split_day`) and `test_df` (data on or after `split_day`).
- Prints the date ranges of the training and testing sets.

10. **Model Training (XGBoost):**

- Defines the feature columns (`feature_cols`) to be used for training the model.
- Separates the features (`X_train`, `X_test`) and the target variable (`y_train`, `y_test`) for both the training and testing sets.
- Initializes and trains an XGBoost Regressor model (`xgb.XGBRegressor`) with specified hyperparameters.
- Makes predictions on the test set (`y_pred`).
- Evaluates the model's performance using Root Mean Squared Error (RMSE).

11. **Result Visualization (Initial Model):**

- Creates a Pandas DataFrame `results_df` containing the actual and predicted quantities for the test set.
- Sorts the results by `item_id` and `order_date`.
- Identifies the top-selling item.
- Plots the actual and predicted daily quantities over time for this top-selling item using `seaborn`.

12. **Feature Engineering - Rolling Average Features:**

- Sorts the DataFrame again by `item_id` and `order_date`.
- Calculates rolling average features for `daily_quantity_sold` over windows of 3 and 7 days, again using `.rolling_mean().over("item_id")` to apply the rolling calculation independently for each item.
- Drops rows with null values resulting from the rolling window calculations.

13. **Model Retraining (with Rolling Features):**

- Prepares the feature columns, now including the rolling average features.
- Re-splits the data into training and testing sets.
- Trains a new XGBoost Regressor model using the expanded feature set.
- Makes predictions and evaluates the performance (RMSE) of the new model.

14. **Result Visualization (Improved Model):**

- Creates a new `results_df` with the predictions from the model trained with rolling average features.
- Plots the actual vs. predicted quantities for the top-selling item using the improved model.

15. **Error Analysis and Labeling:**

- Calculates the percentage deviation between the actual and predicted quantities.
- Defines a function `label_deviation` to categorize the forecast accuracy (e.g., "As Expected", "Better Than Expected", "Worse Than Expected") based on a defined threshold (10%).
- Applies this function to create a `label` column in the results DataFrame.

- Summarizes the count and percentage of predictions in each category.
- Creates and displays a clean summary table showing the forecast date, predicted quantity, actual quantity, deviation percentage, and label.

16. **Incorporating Item ID as a Feature:**

- Uses `LabelEncoder` from `sklearn.preprocessing` to convert the categorical `item_id` into numerical values (`item_id_encoded`).
- Adds `item_id_encoded` to the list of feature columns.
- Re-prepares the training and testing data with the encoded item ID.
- Performs checks for object data types and potential null values in the feature sets, and applies `pd.to_numeric` to ensure all features are numerical.
- Trains a final XGBoost model including the encoded `item_id` feature.
- Evaluates the performance (RMSE) of this final model.
- Visualizes the actual vs. predicted quantities for the top-selling item with the final model.
- Performs error analysis and labeling for the final model's predictions.
- Displays a summary of the forecast accuracy.
- Creates and displays a clean summary table of the daily prediction performance.

17. **Further Data Aggregation and Pivoting (for detailed analysis):**

- Groups the final results by `item_id` and `order_date` to show the total predicted and actual quantities per item per day.
- Creates pivot tables to display the predicted and actual quantities with `forecast_date` as the index and `item_id` as columns, allowing for a side-by-side comparison of forecasts for different items over time.

6.1.1 GrabCast Dataset

To generate the combined view for GrabCast, we performed the following key steps:

1. **Merged transaction_data.csv with transaction_items.csv** to link each order to its items.

   However, during the merging process, we discovered a critical flaw. In multiple cases, the item_id linked to an order_id **did not match the merchant** that fulfilled the order.

   For example:

   - order_id = 46975df1c had merchant_id = 2a1c4.
   - However, the linked item_id = 10 belonged to merchant_id = e7a2f, which is incorrect.
   - This mismatch meant that the item_price used would not reflect the true offerings of the actual merchant, leading to errors in estimating order value and sales forecasts.

   To correct this issue and reconstruct an accurate dataset:

   - We **filtered and reassigned item_ids** for each order, selecting only those items whose merchant_id matched the one from transaction_data.
   - Where mismatches were found, we used a **price approximation technique** to identify a valid combination of item_ids that closely matched the original order_value.
   - This ensured that all item entries per order were consistent with the merchant's catalog. We also validated that the new item combinations were realistic and valid (e.g., subtotal sums matched total order value, quantities were logical).

2. **Joined with items.csv** to retrieve item_price, item_name, and merchant_id for each item (i.item_id = ti.item_id).

3. **Joined with transaction_data.csv** using order_id to bring in:

- order_merchant_id (renamed from td.merchant_id)
- order_value, order_time
- driver_arrival_time, driver_pickup_time, and delivery_time

4. **Joined with merchants.csv** using item_merchant_id = m.merchant_id to get the merchant_name.

5. **Created a computed field** subtotal by calculating item_price × quantity, cast as numeric(10,2), to represent the line-level contribution to order_value.

Data Dictionary

Attribute	**Description**
order_id	Unique identifier of the order
order_merchant_id	Merchant responsible for fulfilling the order
item_id	Unique identifier of the item in the order
item_name	Name of the ordered item
item_price	Price per unit of the item
quantity	Quantity of the item (inferred or calculated)
subtotal	Total cost of this item line (price × quantity)
item_merchant_id	Merchant who owns the item
merchant_name	Name of the merchant (from merchants.csv)
order_value	Total value of the order
order_time	Timestamp when the order was placed
driver_arrival_time	Time when the driver arrived at the merchant

driver_pickup_time	Time when the driver picked up the order
delivery_time	Time when the order was delivered

6.1.2 GrabBack Dataset

To generate the customer insight view for GrabBack, we performed the following key steps:

1. **Filtered transaction_data.csv using merchant_id** to limit the analysis to customers who had previously ordered from a specific merchant.

2. **Joined transaction_data.csv, transaction_items.csv, and items.csv** to build a view named customer_orders, which includes

    - eater_id
    - order_time
    - item_id, item_name
    - merchant_id

3. **Created** last_order by grouping customer_orders by eater_id and selecting the **maximum** order_time as last_order_date to identify the **most recent order** made by each customer for the selected merchant.

4. **Created** favorite_food by grouping the same orders by eater_id and item_name, then counting the number of times each item was ordered. We used ROW_NUMBER() partitioned by eater_id, ordered by COUNT(*) DESC, to retrieve the **most frequently ordered item** per customer.

5. **Joined last_order and favorite_food** to produce the final output,

    - customer_id = eater_id
    - last_order_date
    - favorite_food

    The final dataset was sorted by last_order_date in descending order to prioritize **recently lapsed customers** for re-engagement.

Data Dictionary

Attribute	**Description**
customer_id	Unique identifier of the eater
last_order_date	Timestamp of the most recent order
favorite_food	Most frequently ordered item by the customer from that merchant

6.2 Data Visualisation & Key Insights

6.2.1 Daily Sales Forecasting Revenue (ARIMA)

skimpy summary

Data Summary

Dataframe	Values
Number of rows	24812
Number of columns	14

Data Types

Column Type	Count
string	5
datetime64	4
float64	3
int64	2

number

column	NA	NA %	mean	sd	p0	p25	p50	p75	p100	hist
item_id	0	0	164.3	67.32	109	109	115	264	264	
item_price	0	0	7.58	1.961	4.25	4.25	8.5	9	9	
quantity	0	0	8.852	29.27	1	1	1	3	200	
subtotal	0	0	78.25	263.8	4.25	4.25	8.5	27	1800	
order_value	0	0	122.5	308.1	12	21.61	29.58	42.67	1800	

datetime

column	NA	NA %	first	last	frequency
order_time	0	0	2023-01-01 06:26:00	2023-12-31 22:12:00	None
driver_arrival_time	0	0	2023-01-01 06:45:00	2023-12-31 22:19:00	None
driver_pickup_time	0	0	2023-01-01 06:48:00	2023-12-31 22:25:00	None
delivery_time	0	0	2023-01-01 06:58:00	2023-12-31 22:36:00	None

string

column	NA	NA %	shortest	longest	min	max	chars per row	words per row	total words
order_id	0	0	005d68852	005d68852	000170bb5	fffb2c393	9	1	24812
order_merchant_id	0	0	f7d1a	f7d1a	f7d1a	f7d1a	5	1	24812
item_name	0	0	Pancit Malabon	Pancit Canton with Chicken	Lumpiang Shanghai	Pancit Malabon	18.1	2.6	64638
item_merchant_id	0	0	f7d1a	f7d1a	f7d1a	f7d1a	5	1	24812
merchant_name	0	0	Pancit Hub	Pancit Hub	Pancit Hub	Pancit Hub	10	2	49624

End

Figure 1: Summary Statics for merchant_id "f7d1a"

- The dataset of the merchant with merchant_id “f7d1a” contains **24,812 rows** and **14 columns**, representing item-level food orders from a single merchant (“Pancit Hub”) across 2023.

- The summary statistics also indicate that no missing values were detected in any column.

Key data types:

- **Numerical**: item_price, quantity, order_value, subtotal
- **Datetime**: order_time, pickup_time, delivery_time
- **Categorical**: item_id, item_name, merchant_name
- Most items are priced between **RM 4–9**.
- quantity and order_value are **highly skewed**, indicating potential bulk orders or outliers.
- **Time fields** are complete and suitable for time-series modeling (e.g. forecasting demand, delivery analysis).
- The dataset focuses on a **single merchant**, simplifying modeling without multi-vendor complexity.

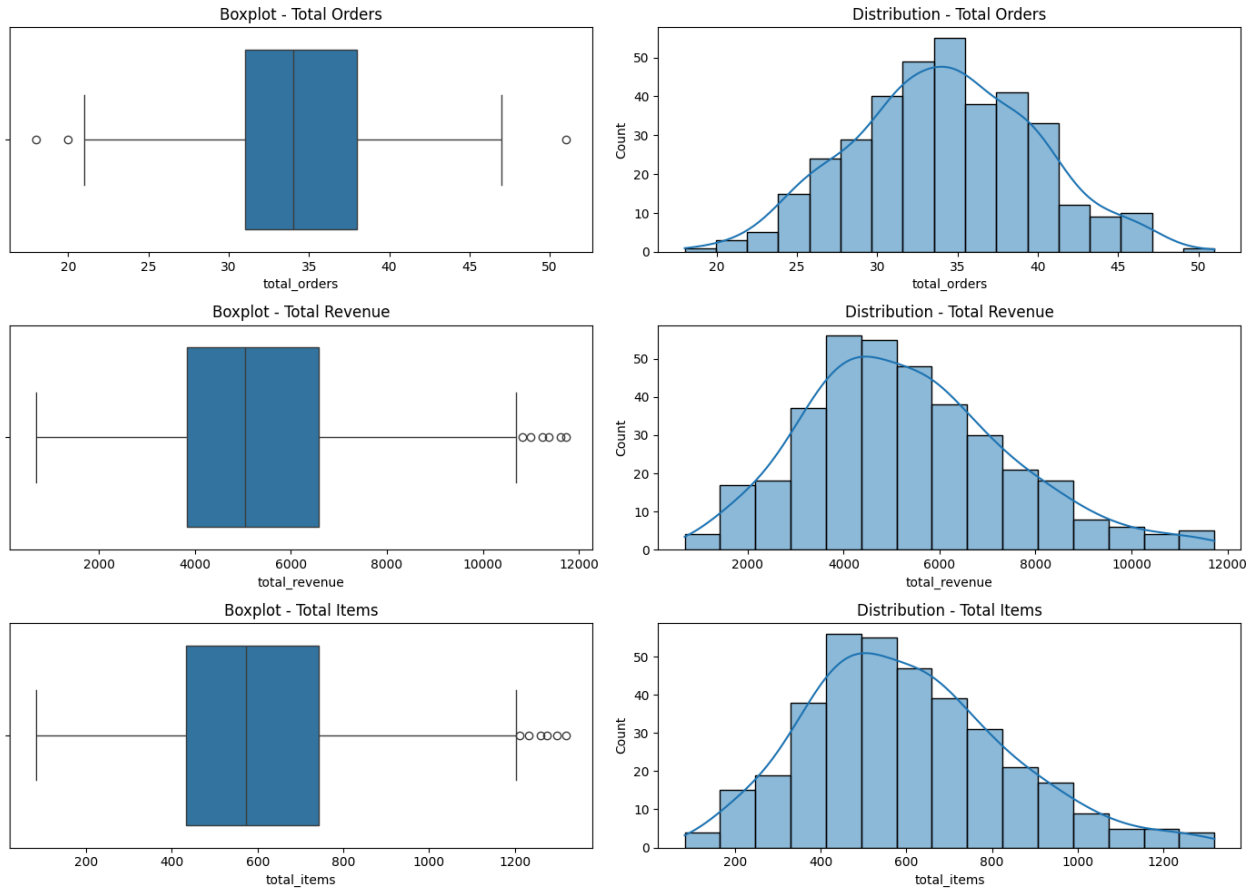


Figure 2: Figure 2: Boxplot (Left) & Histogram with KDE (Right)

1. Total Orders

- **Boxplot**: Most daily **orders lie between 30 and 40**, with a **few low and high-end outliers**.
- **Distribution**: Approximates a **normal distribution**, suggesting consistent order flow with mild variation.

2. Total Revenue

- **Boxplot**: Revenue ranges widely, with noticeable **high-value outliers beyond RM 10,000**.
- **Distribution**: **Right-skewed**, indicating **most days generate RM 3,000–7,000**, but some days **experience sales spikes** due to promotions.

3. Total Items Sold

- **Boxplot**: **Median around 550 items per day**, with **outliers above 1,200**.
- **Distribution**: **Right-skewed**, showing most days fall in the 400–700 range

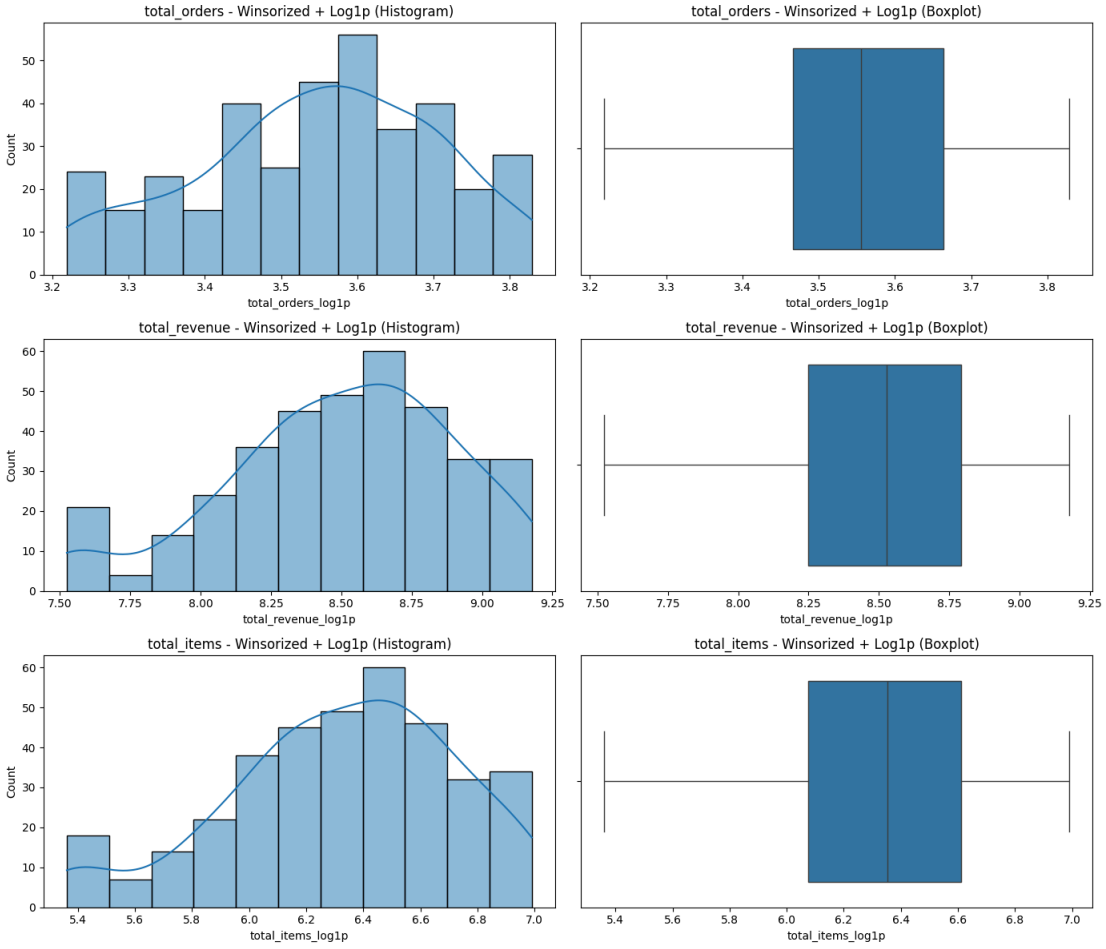


Figure 3: Histogram with KDE (Left) & Boxplot (Right) compares total_orders, total_revenue, total_items before and after Winsorization + Log Transform

1. Total Orders

- **Histogram (Top-Left)**:
  - After transformation, the distribution of total_orders is **fairly symmetric and close to normal**.

    - Slight right skew might still be present, but much reduced from the original scale.

- **Boxplot (Top-Right)**:

    - Central box shows **low variance**, with no extreme outliers after winsorization.

    - Data is well-contained within whiskers — suggesting effective outlier mitigation.

2. Total Revenue

- **Histogram (Middle-Left)**:

    - Distribution is **bell-shaped and smooth**, indicating a good normalization result post-log transformation.

    - Still slightly skewed, but very manageable.

- **Boxplot (Middle-Right)**:

    - Most values are within a tight interquartile range.

    - No strong presence of outliers, and the distribution appears centered.

3. Total Items

- **Histogram (Bottom-Left)**:

    - Distribution also appears **symmetrical with a slight right skew**, more normalized than raw values.

    - Peak around log1p ≈ 6.4 suggests this is a common transaction size (in item count).

- **Boxplot (Bottom-Right)**:

    - Compact IQR with minimal whiskers — indicating reduced variance and few/no outliers post-winsorization.
    - Centered distribution implies a healthy median.

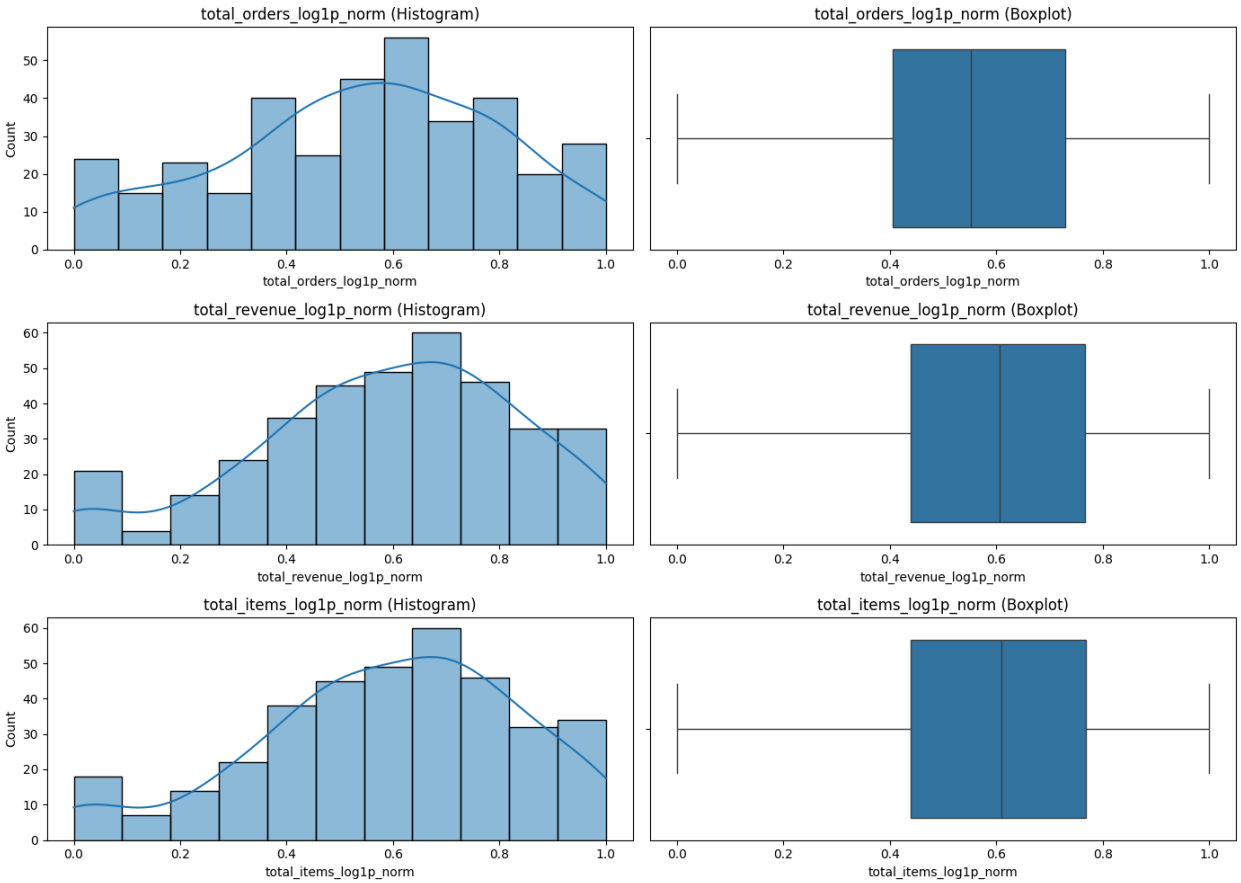


Figure 4: Histogram with KDE (Left) & Boxplot (Right) isualizing Min-Max Normalized Features

These histograms and boxplots illustrate the distribution and spread of the three normalized variables after applying Log1p transformation followed by Min-Max scaling to compress the range into [0, 1].

1. Total orders

- **Histogram**: Displays a moderately symmetric distribution with values centered around 0.6. The bell-like shape suggests effective transformation and scaling.

- **Boxplot**: Data is well-centered with no significant outliers. The IQR (interquartile range) is narrow, showing consistent scaling.

The feature is now balanced and free from extreme values, making it suitable for machine learning models that are sensitive to scale.

2. total_revenue_log1p_norm

- **Histogram**: Slight right skew, but the distribution is mostly smooth and unimodal.
- **Boxplot**: A symmetric and clean spread of data within [0, 1], no apparent outliers or extreme values.

Revenue values have been normalized effectively, making this feature ready for modeling without risk of scale-related distortion.

3. total_items_log1p_norm

- **Histogram**: Shows a well-shaped distribution, peaking around 0.6 with a gentle slope toward both extremes.
- **Boxplot**: Compact and centered box with no major outliers; indicates clean and consistent normalization.

The distribution is ideal for downstream tasks such as regression or clustering due to its even spread and lack of outlier influence.

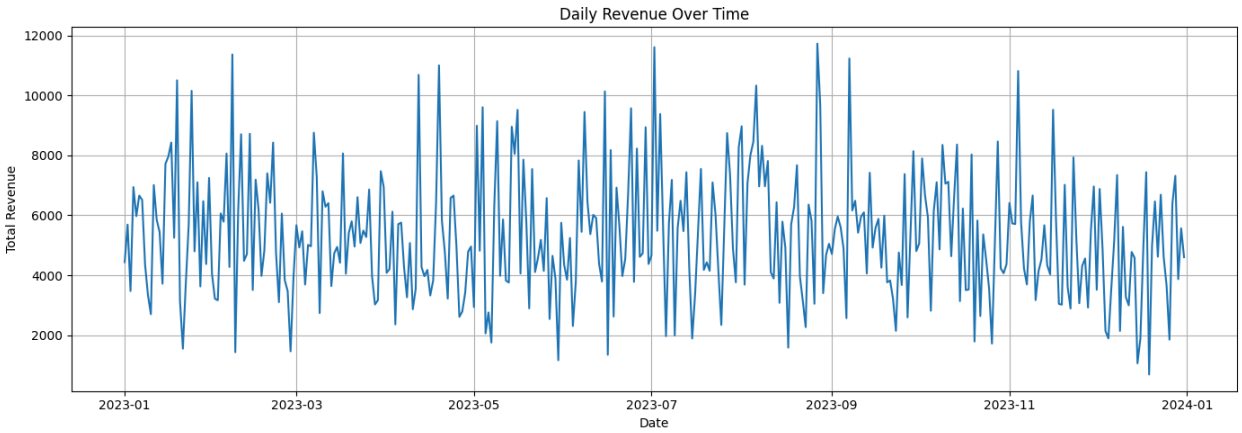


Figure 5 : Daily Revenue Over Time

- The plot visualizes **total revenue** generated **per day** throughout the year **2023**.
- The x-axis represents the **date**, while the y-axis shows the **daily revenue in monetary units**.

1. **High Volatility**
    - The revenue fluctuates significantly from day to day.
    - There are frequent sharp spikes and drops, indicating **inconsistent sales volume**.
2. **Periodic Spikes**
    - Several days show **extremely high revenue peaks**, likely due to promotional events, peak meal hours, weekends, or external campaigns.
3. **Lack of Clear Trend**
    - No obvious long-term upward or downward trend is visible over time.
    - This suggests **revenue is more seasonal or event-driven** rather than steadily increasing or declining.

4. **Presence of Noise**

    - The raw revenue data is quite **noisy**, justifying the need for smoothing techniques like:

        - Rolling averages
        - Log transformations
        - Seasonal decomposition

This plot highlights the **importance of preprocessing and transformation** before modeling. It reinforces the need for:

- **Stationarity testing** (for ARIMA)
- **Trend + seasonality analysis**
- **Outlier suppression** or smoothing for more stable forecasting

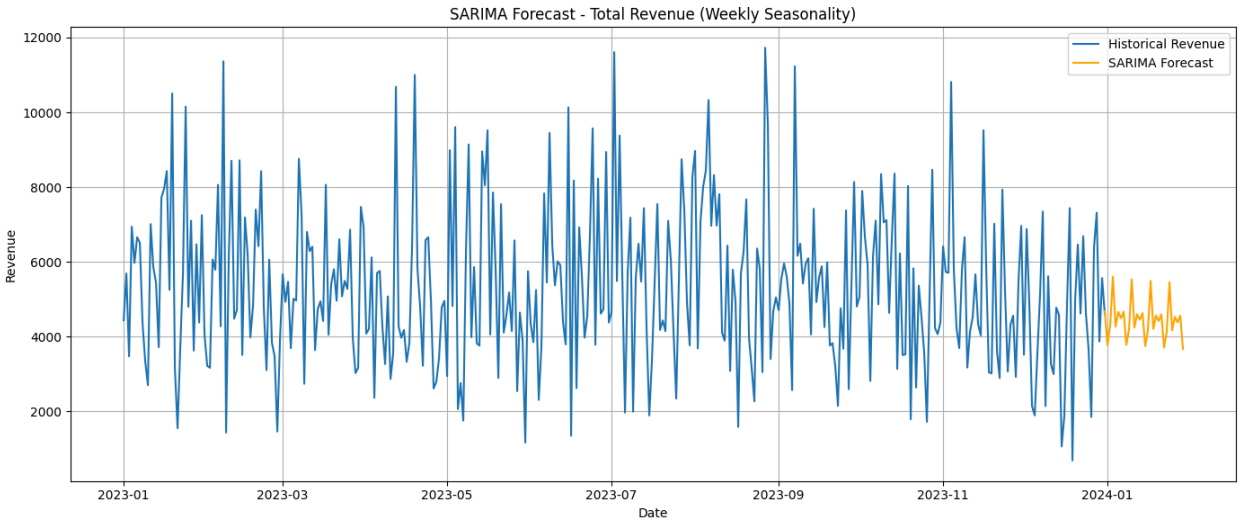


Figure 6: SARIMA Forecast – Total Revenue (Weekly Seasonality)

- **Blue Line**: Historical daily total revenue for the year 2023.

- **Orange Line**: SARIMA model's **forecasted revenue** for a future period (e.g., next 30 days).

- The model incorporates **weekly seasonality** — denoted by the repeated short-term cycles in the forecast.

1. **Model Fit and Forecast Extension**
    - The model was trained on 2023 data and forecasted into early 2024.
    - Forecasted values continue the expected pattern seen in past trends, capturing weekly ups and downs.
2. **Seasonal Pattern Captured**
    - The orange forecast line shows regular **short-term fluctuations**, consistent with weekly demand cycles (e.g., weekdays vs weekends).
    - Indicates that the SARIMA model is appropriately incorporating **seasonal structure (s=7)** in the time series.
3. **Stability of Forecast**
    - Forecasted values remain within a **narrow range** compared to the higher variance seen in historical data.
    - This is typical when noise is smoothed and the model predicts central tendencies.
4. **Realistic Prediction Behavior**
    - No overfitting is observed in the forecast: the output doesn't swing unrealistically high or low.
    - Suggests the SARIMA configuration is robust (likely with parameters like (p,d,q)x(P,D,Q,s)).

The SARIMA model provides a stable and realistic short-term revenue forecast, accounting for weekly cyclic behavior.

Useful for **inventory, staffing, and marketing** decisions by giving merchants a data-driven look at expected revenue patterns.

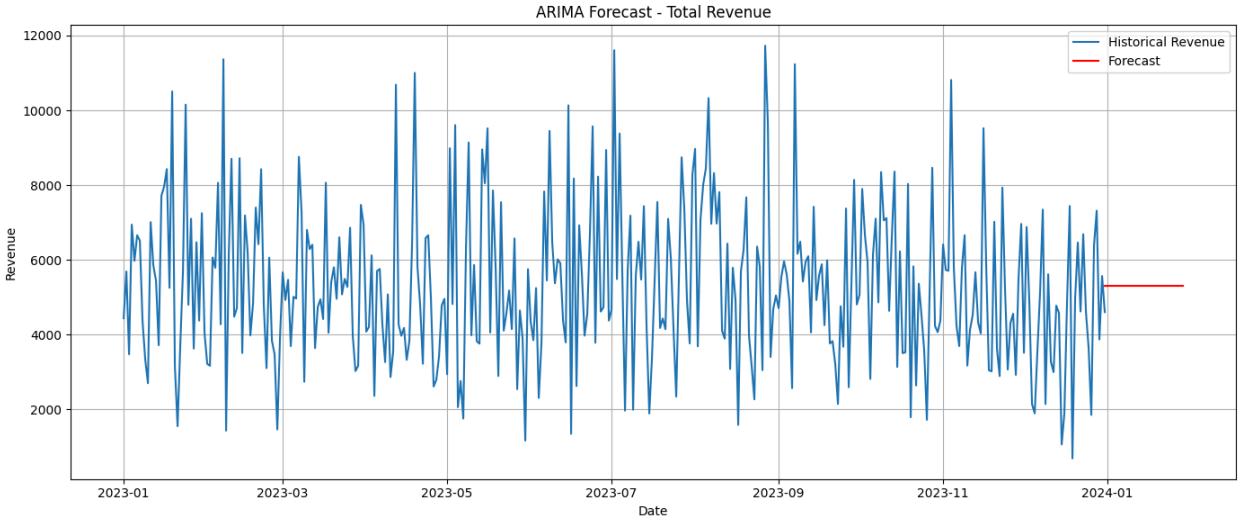


Figure 7 : ARIMA Forecast (Total Revenue)

- Blue Line: Historical daily total revenue for the year 2023.
- Red Line: ARIMA model's forecast for future daily revenue (short-term horizon, e.g., 30 days).
- The forecasted values appear as a flat line, indicating a constant average-level prediction.

1. **No Seasonality Modeled**

    - The forecast line is **flat**, which is typical of standard ARIMA models without seasonal components.

    - ARIMA captures **trend and autocorrelation**, but not repeating patterns (e.g., weekdays vs weekends), unlike SARIMA.

2. **Forecast Centers Around Mean**

- The red line suggests the model expects future revenue to hover near the **historical average**.

- This implies that ARIMA detected **no clear upward or downward trend** in the differenced data.

3. **Low Forecast Variance**

    - Unlike real-world revenue behavior (which is volatile), the forecast is **overly smooth**.

    - This may lead to underestimation or overestimation of future spikes or dips in revenue.

4. **Suitable for Stationary Series**

- ARIMA is more appropriate when the time series is made stationary and lacks strong seasonal cycles.

- The result here reflects a model that has been trained on noise-reduced, differenced data with limited pattern repetition.

The ARIMA model provides a **basic average-level forecast** that works best for **short-term, non-seasonal predictions**.

- It lacks the cyclical behavior seen in more advanced models (e.g., SARIMA), making it less suitable if weekly seasonality is present in the data.

- Useful as a baseline model, but may benefit from switching to SARIMA or Prophet if seasonality is a key factor.

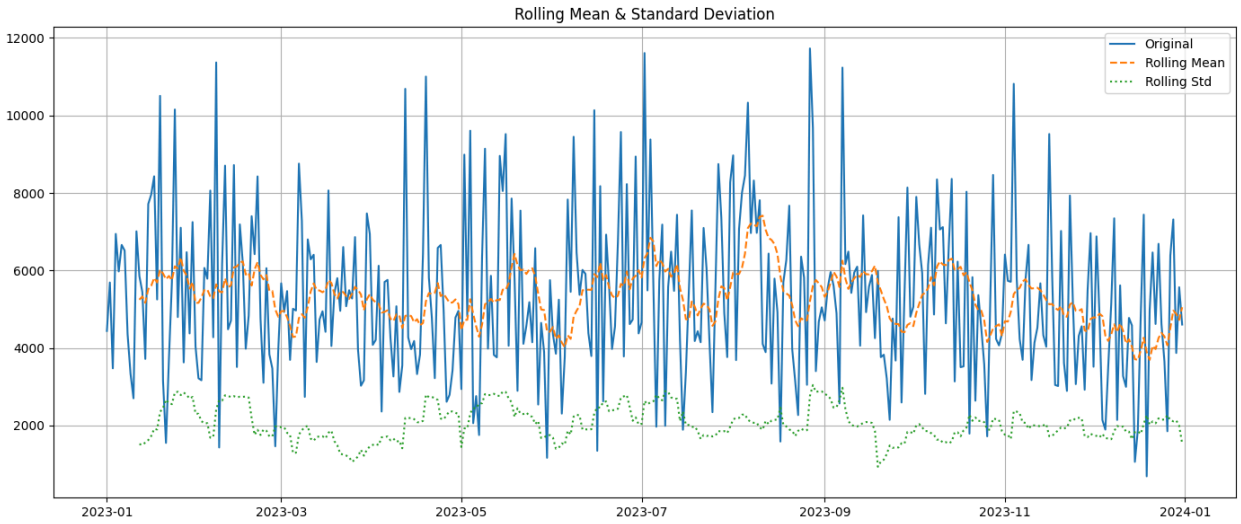


Figure 8: Rolling Mean & Standard Deviation

- **Blue Line**: Original daily revenue data (2023).
- **Orange Dashed Line**: 12-period **rolling mean** (trend over time).
- **Green Dotted Line**: 12-period **rolling standard deviation** (volatility over time).

1. **Fluctuating Rolling Mean**
    - The **rolling mean is not flat**; it shows cyclical up-and-down movement.
    - Indicates the presence of **non-stationarity in the mean** — revenue patterns change over time.
2. **Changing Rolling Standard Deviation**

- The **rolling std** also varies significantly, especially during certain months.
- Confirms that **variance is not constant**, another sign of non-stationarity.

3. **Strong Seasonality Patterns**
    - Repeated waves in both mean and std lines suggest **seasonal trends**, likely influenced by day-of-week or promotional cycles.

4. **Need for Differencing**

- Because both the rolling mean and std dev fluctuate, the time series violates the assumptions of stationarity required for ARIMA.
- This motivates the use of:
    - **Differencing** (to remove trend)
    - **Seasonal differencing** (if using SARIMA)
    - **Log transformation** (to stabilize variance)

The plot clearly shows that the **original revenue series is non-stationary**. Rolling mean and std dev are not stable over time.

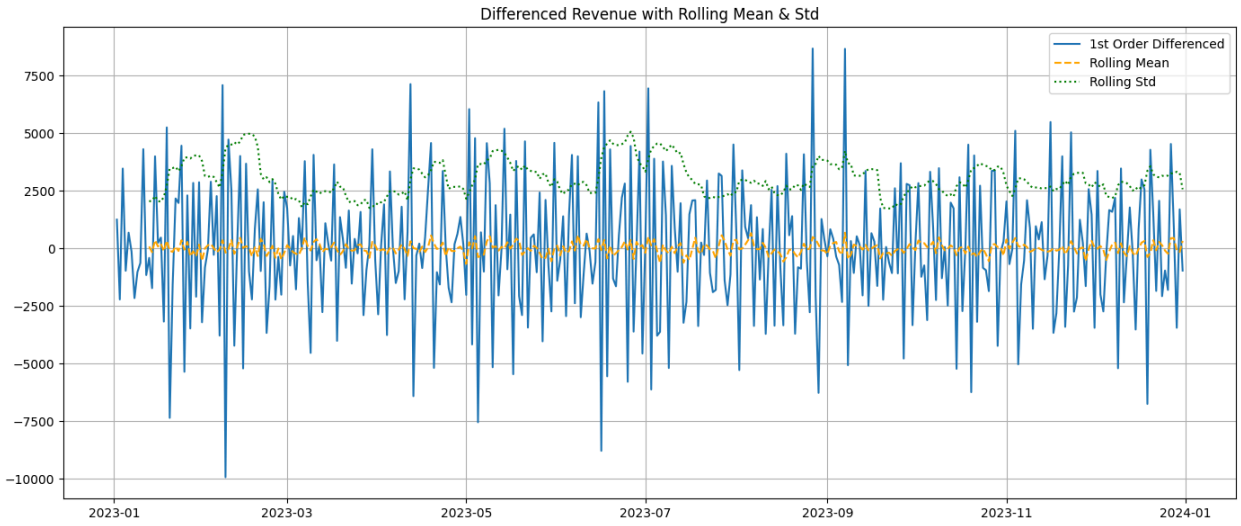


Figure 9: Differenced Revenue with Rolling Mean & Standard Deviation

- **Blue Line**: First-order differenced daily revenue (`revenue_t - revenue_t-1`)
- **Orange Dashed Line**: Rolling mean of the differenced series (trend check)

- **Green Dotted Line**: Rolling standard deviation of the differenced series (volatility check)

1. Rolling Mean is Nearly Flat

- The orange line hovers closely around zero throughout the year.
- Indicates that the **trend component has been removed**, addressing non-stationarity in the mean.

2. Rolling Standard Deviation is Relatively Stable

- The green line shows less variation compared to the original series (before differencing).
- Suggests that **variance is now more consistent over time**, a sign of improved stationarity.

3. Series Still Noisy but Statistically Stationary

- Although the differenced series (blue line) looks noisy, that's expected.
- What matters for ARIMA is that the mean and variance don't drift and here they don't.

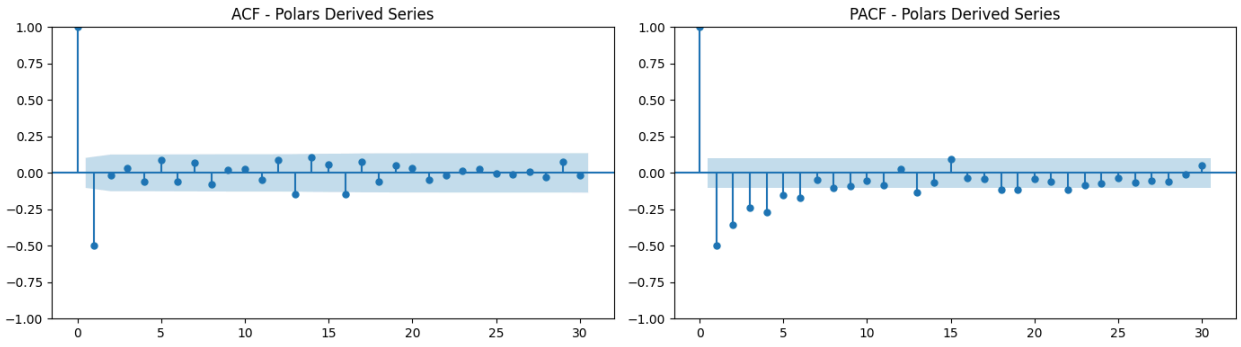


Figure 10: ACF & PACF Plots (After Differencing)

- **Left Plot: ACF (Autocorrelation Function)**
    - **Lag 1 shows a strong negative autocorrelation**, dropping sharply below the significance boundary.
    - Remaining lags hover within the confidence interval (blue shaded area), suggesting **no significant autocorrelation** beyond lag 1.
    - This indicates that **q = 1** (MA component of ARIMA) may be sufficient.

- **Right Plot: PACF (Partial Autocorrelation Function)**
    - **Lag 1 again shows a significant drop** (strong negative partial autocorrelation).
    - Several minor values outside the confidence band at lags 2–4, then mostly decay within bounds.
    - This suggests a good fit with **p = 1**, possibly up to **p = 2 or 3** depending on model performance.

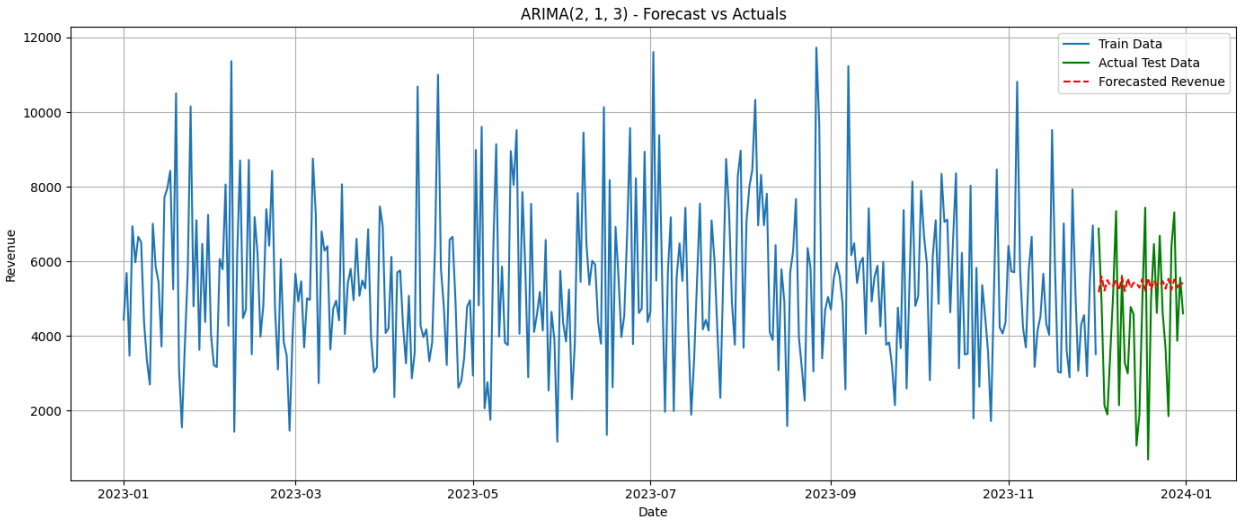


Figure 11: ARIMA(2,1,3) Forecast vs Actual Revenue

- **Blue Line**: Historical training data for total revenue (2023).
- **Green Line**: Actual revenue values from the **test set**.
- **Red Dashed Line**: **Forecasted revenue** generated by the ARIMA(2,1,3) model for the test period.

1. **Model Type**
    - ARIMA(2,1,3) indicates:
        - **p=2**: Includes 2 lags of autoregression
        - **d=1**: First-order differencing (to achieve stationarity)
        - **q=3**: Includes 3 lags of moving average
2. **Prediction Behavior**
    - The **forecasted values (red)** are relatively flat and consistent across the prediction window.

- This is a common behavior of ARIMA models trained on differenced series without explicit seasonal components.

3. **Gap Between Actual vs Forecast**

   - Some **visible deviation** exists between actual (green) and predicted (red), especially during revenue peaks and dips.

   - Suggests the ARIMA model captures general trend/level, but struggles with **volatility and sharp changes**.

4. **Bias Toward Historical Mean**

   - The predicted values tend to hover near the **average level** of the recent training data.

   - This reinforces the **lack of seasonality modeling**, limiting the model's adaptability to real-world fluctuations.

ARIMA(2,1,3) provides a **stable, mean-reverting forecast** that captures the general trend but **misses high-frequency fluctuations** in test data.

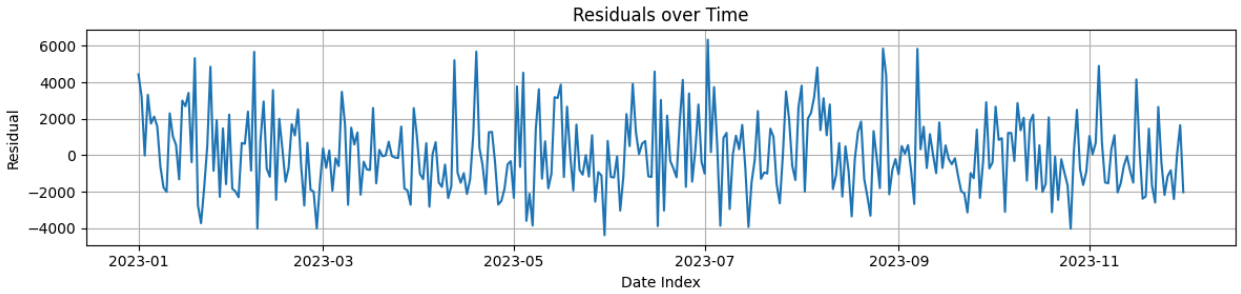


Figure 12: Residual Line Plot (Time Series)

- The plot of residuals over time shows **no clear pattern or trend**.

- Residuals fluctuate randomly around zero, which is a good sign.

- Suggests that the ARIMA model has effectively removed temporal structure from the data.

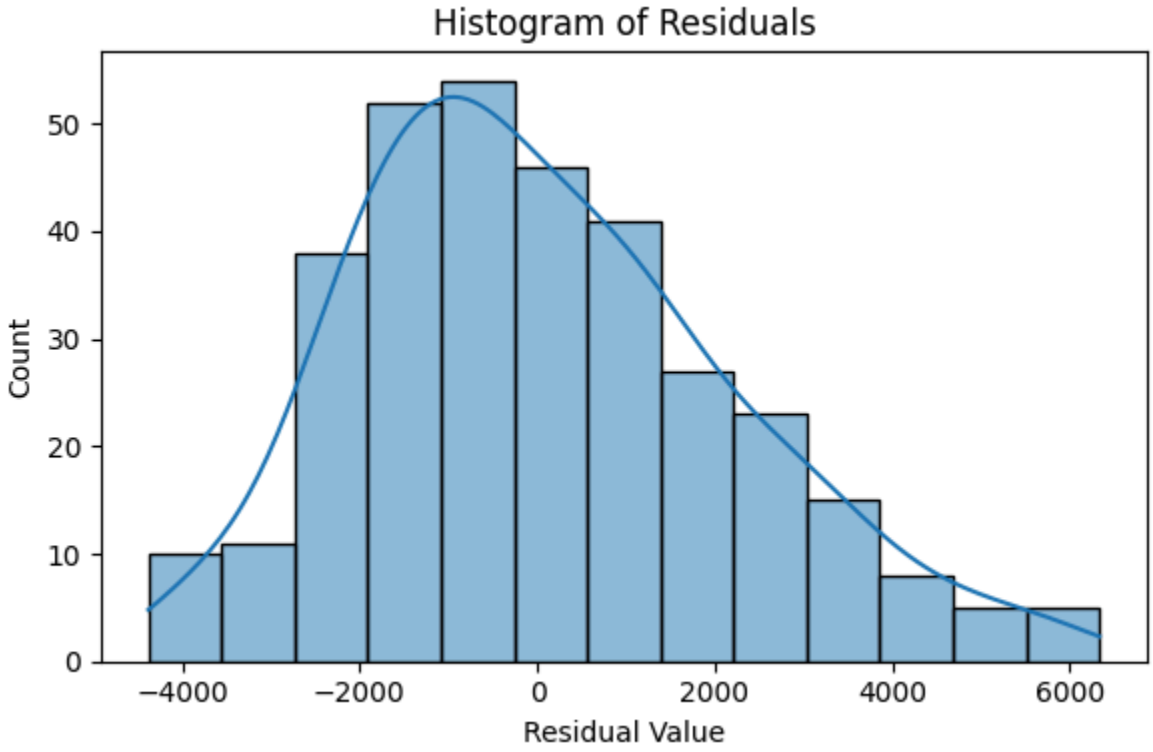


Figure 13: Histogram with KDE (Residual Distribution)

- The histogram resembles a **bell-shaped curve**, centered around zero.
- The KDE overlay confirms a **roughly normal distribution** of residuals.
- Indicates the residuals are symmetrically distributed and meet the Gaussian noise assumption.

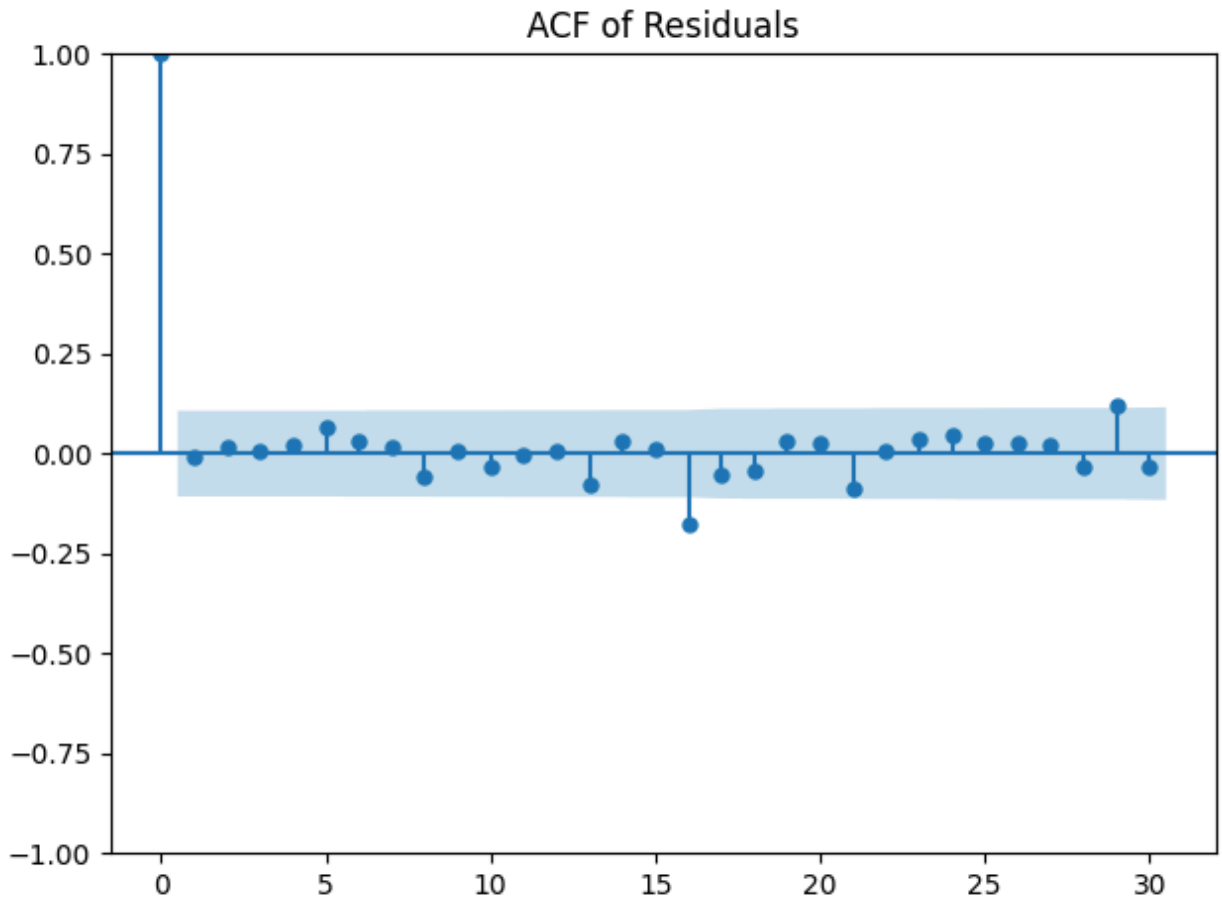


Figure 14: ACF Plot (Residual Autocorrelation)

- Most spikes fall within the 95% confidence interval (shaded blue band).
- No significant lags indicate the residuals are **not autocorrelated**.
- Suggests the model does **not leave systematic patterns** unexplained.

6.2.2 Item Quantity Prediction (using XGBoost)

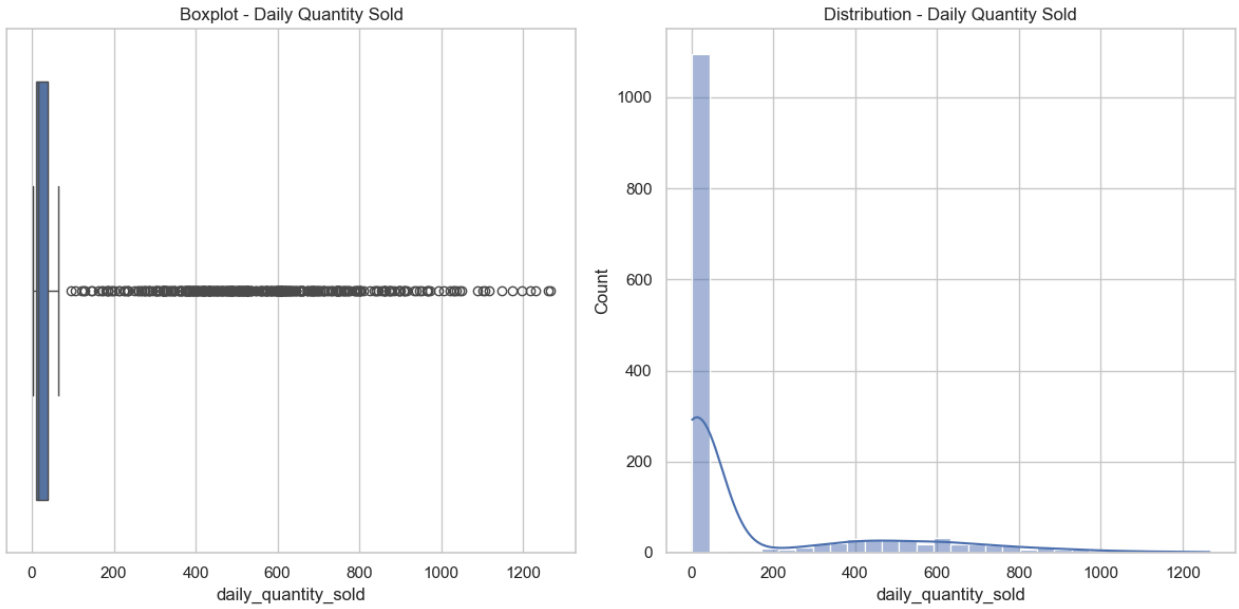


Figure 15: Boxplot (Left) & Histogram with KDE (Right)

We analyzed the distribution of daily_quantity_sold across all items using two visualizations:

Box Plot

- The boxplot highlights the **spread and presence of outliers** in daily item sales.
- The median daily sales is relatively low, with the IQR tightly packed around lower values.
- Numerous extreme outliers are observed, with values extending beyond 1,200 units in a single day and likely due to promotions or bulk orders.

Histogram with Kernel Density Estimation (KDE)

- Shows the **frequency distribution** of daily item sales.
- Most items sell **fewer than 100 units per day**, forming a sharp left-skewed distribution.
- A **long right tail** confirms that only a few days experience extremely high quantities sold.

The **distribution is heavily skewed**, making normalization or log-scaling a potential consideration in modeling.

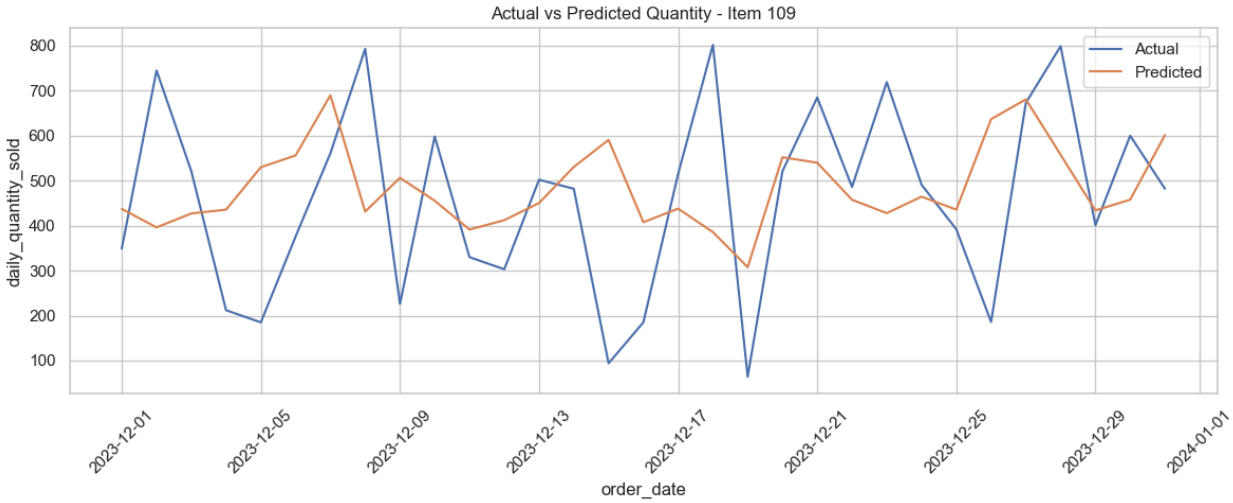


Figure 16: XGBoost Forecast – Daily Quantity Sold (Item 109)

- **Blue Line (Actual)**: Real historical values of `daily_quantity_sold` for Item 109 from early December 2023 to early January 2024.
- **Orange Line (Predicted)**: Quantity values predicted by the trained **XGBoost model**.

1. **Trend Alignment**
    - The predicted line generally follows the **overall direction** of the actual line.
    - Peaks and troughs are aligned, though the predicted values appear **smoother and less volatile**.
2. **Underestimation of Spikes**
    - The XGBoost model tends to **underestimate high spikes** in actual quantity (e.g., around Dec 3, 10, 17, 25).

- Suggests the model may be **averaging out extreme sales surges**, possibly due to limited feature capture or noise suppression.

3. **Over-smoothing**

    - While the predicted values avoid erratic jumps, they **fail to capture day-to-day variance** accurately.

    - This is common in models that prioritize low RMSE over peak accuracy.

4. **Stable Performance**

    - Despite missing peaks, the forecast line consistently stays within a **reasonable margin** from the actual values.

    - Indicates a **stable and generalizable model**, which is useful for trend monitoring and inventory planning.

The XGBoost model delivers a **reasonably accurate daily forecast** of item quantity sold. It captures **seasonal patterns** and general flow but **misses sharp spikes**

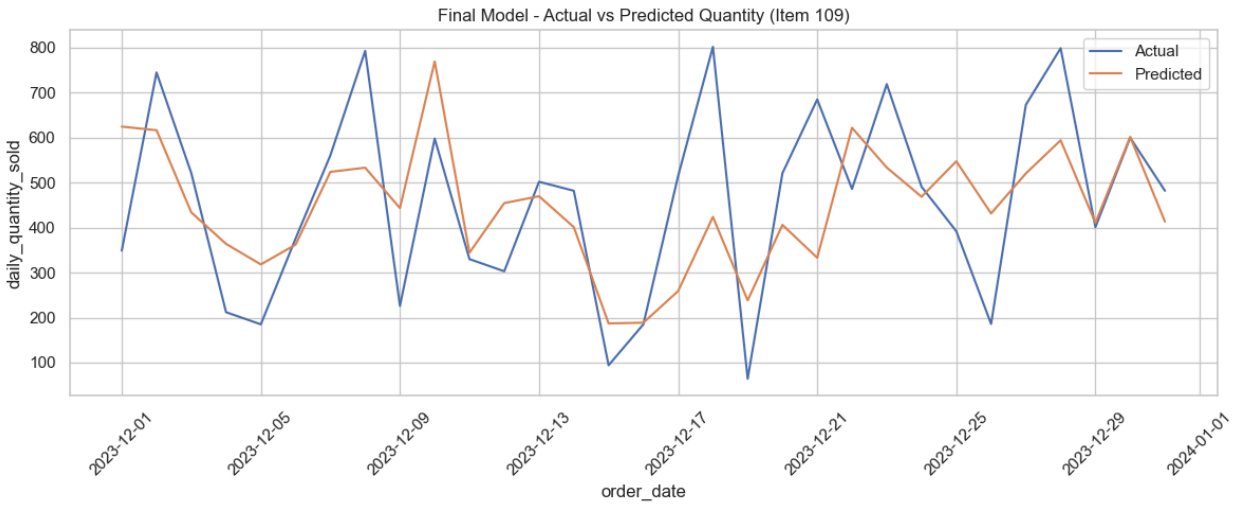


Figure 17: XGBoost model with both `item_id` and rolling averages

1. **Improved Pattern Recognition**
    - The predicted line (orange) more closely follows the general trajectory of the actual line (blue), especially during mid to late December.

    - This suggests the model has learned **short-term temporal dynamics** using rolling averages.

2. **Item-Specific Learning via Encoding**

    - Incorporating `item_id` allowed the model to tailor its understanding to Item 109's unique sales pattern, rather than applying a generic rule to all items.

3. **Better Forecast Precision**

    - The model captures **multiple peaks** (e.g., Dec 9, Dec 21, Dec 27–30) and **smoother transitions** in daily demand.

    - Some shortfalls remain in areas of high variance (e.g., Dec 17–19), but these are less severe compared to earlier iterations.

4. **Reduced Under/Overfitting**

    - Predictions are neither overly smoothed nor excessively reactive, achieving a **balance between variance and bias**.

By integrating **rolling averages** and **item-level encoding**, the XGBoost model significantly improved its predictive accuracy. The visual shows that it is now more responsive to **individual product demand trends**

6.3 Tech Stack (Technical Execution)

This project leverages a modern, efficient stack chosen for rapid development and potential scalability:

- **Frontend: React** with **Mantine UI** - Provides a fast, component-based development experience for a responsive user interface
  .
- **Backend: FastAPI (Python)** - High-performance asynchronous framework ideal for building APIs quickly and efficiently.

- **AI Core: Ollama** with **Llama 3** - Enables local execution of powerful Large Language Models, simplifying setup and ensuring data privacy during development/demos. Feasible for initial deployment and scalable by migrating Ollama to dedicated hardware.

- **Prototyping: Figma** - Used for initial UI/UX design and prototyping (Link to Prototype).

- **(Optional) Storage:** Utilizes lightweight **CSV files** for initial data handling, with potential to integrate **PostgreSQL** for robust, scalable data persistence.

This stack demonstrates **simplicity** (local LLM, standard web tech), **feasibility** (rapid development frameworks), and **scalability** (FastAPI performance, potential for dedicated AI/DB resources).

7.0 How It Works (Feasibility & Communication)

The system operates through a clear, modular flow:

User Interaction (Frontend): The merchant types a request or goal into the React/Mantine chat interface.

Intent Analysis (Backend - /API/chat):

- The message is sent to the FastAPI backend.
- An initial call is made to the local Ollama (Llama 3) instance using a structured prompt (prompt1.txt) with few-shot examples.
- Ollama analyzes the text and extracts key intents related to Sales Timing/Performance and Discount Timing/Strategy.

Response Formatting (Backend - Python Logic):

- Simple Python logic checks which intents were detected (Sales only, Discount only, Both, or None).
- The corresponding response formatting function (format_..._response) is called.
- This function prepares a user-friendly message acknowledging the intent and suggests relevant next actions (e.g., "Generate Sales Forecast", "Draft Retention Messages").

Display & Suggest Actions (Frontend): The backend sends the formatted message and suggested action list back to the React frontend, which displays them to the merchant.

Action Execution (Backend - /api/execute_action):

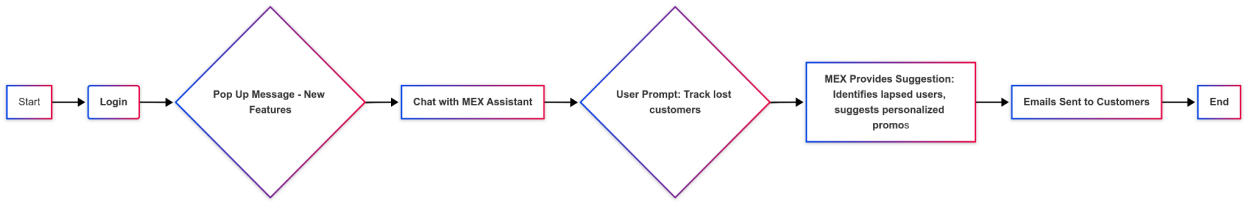
- If the merchant clicks a suggested action button, the frontend sends the action name to a dedicated FastAPI endpoint (/api/execute_action).
- The backend maps this name to the appropriate internal Python function (generate_sales_forecast_logic, etc.) and executes it (currently placeholders, but designed for real logic).
- The result/confirmation message from the executed action is sent back to the frontend.

Display Result (Frontend): The frontend displays the outcome of the executed action as a new message from the bot.

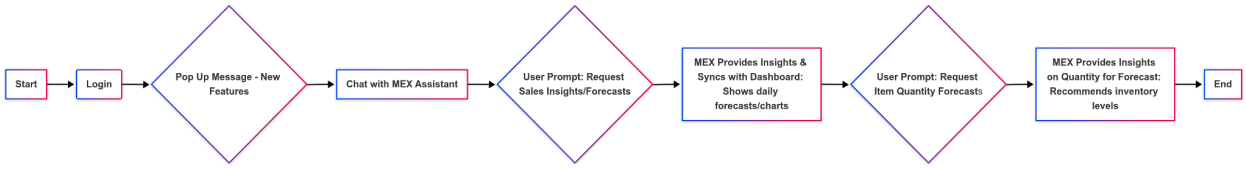
This demonstrates feasibility through its modular design (separate endpoints for chat and actions), clear communication between components via simple JSON APIs, and leverages the local LLM efficiently for the core intent understanding task.

7.1 Activity Diagram for GrabBack and GrabCast

GrabBack



GrabCast



8.0 Demo Video & Relevant Links

Watch Demo Functional Prototype Here: UM Hack
Proposed Prototype (Figma):
https://github.com/user-attachments/assets/85ec0d68-355d-49d9-b0b7-d9b2d5692548

9.0 Long-Term Impact & Key Recommendations

From AI Tools to a CRM-lite Ecosystem

GrabBack and GrabCast serve as foundational components of a future **CRM-lite system** for Grab merchants. By combining smart customer engagement and sales forecasting, these tools can evolve into an integrated dashboard that empowers merchants to manage their entire customer lifecycle — from acquisition to retention — within the Grab platform. Over time, new modules can be layered on, including customer segmentation, loyalty tracking, multi-channel promotions, and deeper behavioral analytics.

Key Future Features & Integrations

SMS / WhatsApp Integration

- Allow merchants to send personalized promo messages directly via **SMS or WhatsApp**, extending beyond in-app notifications.
- Ensures wider reach to customers who may not open the Grab app regularly, especially in markets where WhatsApp is the dominant communication tool.
- Can be automated to trigger based on user behavior (e.g., no order in 14 days).

Automated Campaign Scheduling

- Enable merchants to set up **"set-and-forget"** campaigns with predefined triggers (e.g., birthday offers, 1-month no order, festival promotions).
- Uses AI to recommend optimal send times and promo types based on customer behavior.

A/B Testing and Optimization

- Merchants can test multiple promo messages and formats to determine what performs best.
- AI can analyze engagement and conversion results to automatically select the best-performing version.
- Over time, the system learns which type of messaging works best per customer segment.

AI-Based Customer Segmentation

- Move beyond generic re-engagement and create **hyper-targeted campaigns** for high-value, price-sensitive, or loyal users.
- Boosts marketing efficiency and ROI, even for merchants with no marketing background.

Integration with Inventory Systems

- Combine GrabCast insights with inventory tracking to provide **auto-restocking recommendations**.

- Enables merchants to transition toward zero-waste operations by syncing demand forecasting with supply planning.

Strategic Benefits to Grab and Merchants

For Grab

- Strengthens merchant stickiness and loyalty to the Grab platform by becoming their go-to business toolkit.

- Encourages deeper use of merchant tools, driving **platform engagement** and **data generation**.

- Enhances Grab's reputation as a **merchant-first, AI-forward platform**, especially in SEA markets where digital tools are still being adopted.

- Opens up new monetization opportunities via premium features or "CRM Pro" tiers.

For Merchants

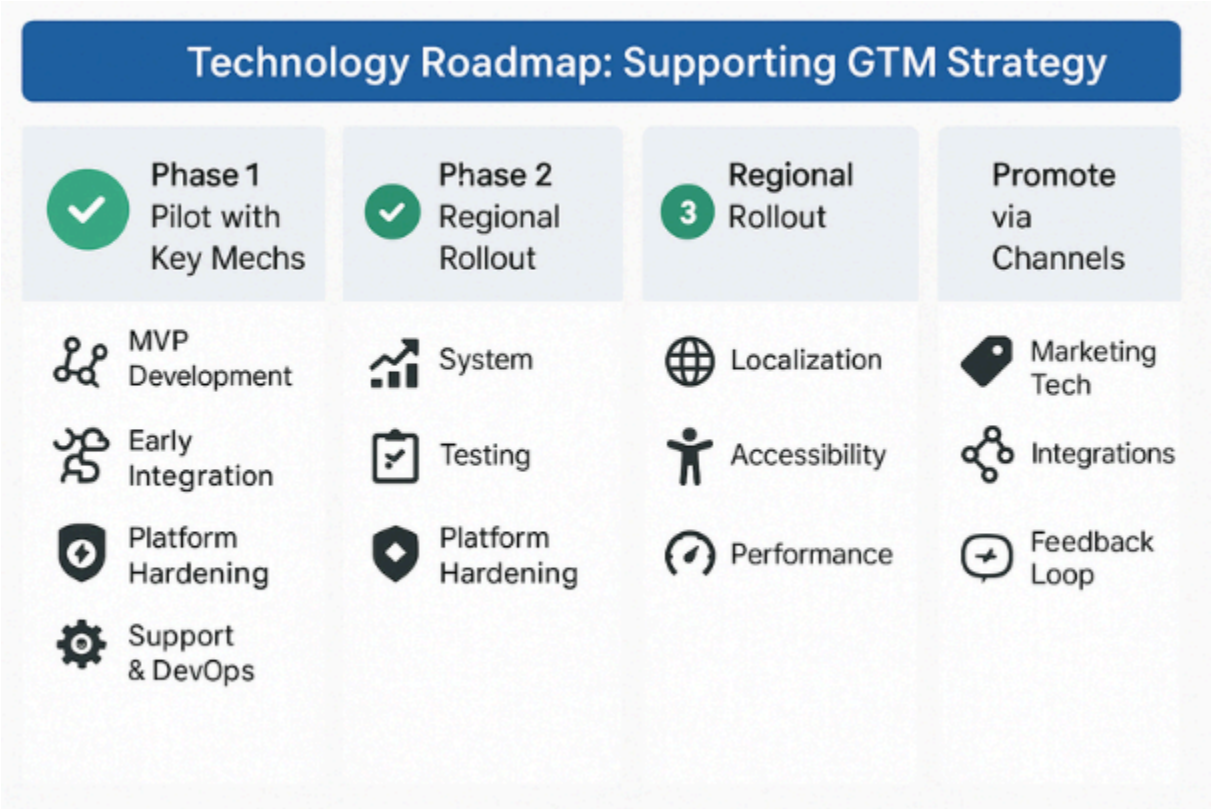
- Offers a scalable way to **grow their business sustainably** without needing technical or marketing skills.

- Reduces churn, increases sales consistency, and enhances inventory efficiency.

- Empowers micro and small businesses to operate like larger enterprises with minimal cost.

- Helps merchants achieve **long-term profitability and resilience**, especially in competitive food and beverage sectors.

9.1 Technology Roadmap for our GrabBack & GrabCast



10.0 Limitations & Suggestions

GrabBack:

- **Dependent on Historical Data:** The system relies heavily on past transaction records to generate personalised offers. In cases where data is sparse or incomplete, recommendations may be less relevant or impactful.

- **Privacy & Consent:** Automated targeting of lapsed customers based on order history requires careful handling of customer data and compliance with privacy regulations (e.g., PDPA).

Suggestion:

- **Privacy-First Design**: Implement opt-in prompts for customers to receive personalized re-engagement messages (PDPA complaint).

- **Smart Defaults**: For low-data users, start with broader category-based offers and personalize over time.

- **Merchant Feedback Loop**: Let merchants rate promo success to improve future targeting.

- **A/B Testing Tool**: Include built-in testing to help merchants compare promo formats with ease.

GrabCast:

- **Accuracy Limitations:** Predictions may not be fully reliable in the absence of sufficient historical data, especially for new merchants or those with inconsistent order patterns.

- **External Factors:** Sales forecasts may not account for sudden external events like public holidays, weather changes, or viral trends that significantly affect demand.

- **Cold Start Problem:** Newly onboarded merchants may experience low forecast accuracy initially due to limited data for model training.

Suggestion:

- **Loyalty-Based Access:** Offer GrabCast to loyal merchants (e.g., 6+ months on platform) to ensure sufficient data for accurate forecasting.

- **Event Tagging:** Allow merchants to tag special days (e.g., festivals, promotions) to improve future forecasts.

- **External Data Integration:** Gradually incorporate external variables (e.g., weather, public holidays) to refine predictions.

- **Guided Cold Start:** For new merchants, start with industry benchmarks and refine as data builds up.

11.0 Team & Roles

Name	Role	Responsibilities
Kaviraj Vijayanthiran 	**Preliminary Round:** Team Lead & AI/ML Developer **Final Round:** Team Lead & AI/ML Developer	**Preliminary Round:** Led project direction and planning, developed machine learning models for forecasting and personalization, integrated AI logic with backend components. **Final Round:** Developed GrabBack and led final implementation of AI features, ensured technical feasibility and team coordination.
Angel Tan Xian Theng 	**Preliminary Round:** Project Coordinator & Business Strategist **Final Round:** Business Strategist & Data Analyst	**Preliminary Round:** In charge of project documentation, concept development, and pitching. Responsible for strategy and project direction. Ensures project aligns with business goals and communicates vision to stakeholders. **Final Round:** Led pitching and flow of the presentation, focused on data storytelling and provided strategic input for

		EDA and business relevance.
Lim Shen Yik 	**Preliminary Round:** Frontend Developer & UI Designer **Final Round:** Data Engineer & Data Analyst	**Preliminary Round:** Designed the user interface using Figma, built frontend components for the prototype, ensured user-friendly experience and clean visual design. **Final Round:** Conducted exploratory data analysis (EDA), processed and transformed datasets for AI modeling, supported backend data workflows.
Siew Jun Zhen 	**Preliminary Round:** Frontend Developer & AI/ML Developer **Final Round:** AI/ML Developer & Data Engineer	**Preliminary Round:** Assisted in building the user interface, contributed to the implementation of AI features on the frontend, and supported ML model integration. **Final Round:** Developed GrabCast, including data pipeline setup and ML model for sales forecasting.

12.0 Working Process

We followed an **agile-inspired approach** over the hackathon timeline with short check-ins and adaptive planning:

- **Daily Standups**: 5 hour brainstorm & working process via Discord.

- **Task Division**: Tasks were divided based on expertise using a shared Google Docs.

- **Review & Syncs**: Regular sync-ups for integration testing and ensuring design-to-logic consistency.

13.0 Task Management Board

We used **Google Docs** to manage and visualize our task flow. Below is a snapshot of our task board:

Task	Notes	PIC	Deadline	Status
BRAINSTORMING		ALL	***8/4/25***	Completed
Documentation outlining solution architecture, data utilisation and personalisation)		Angel Shen Yik	11th April (5pm)	In Progress
Figma Prototype		Darren Shen Yik	11th April (10 pm)	In Progress
Functional Chatbot		Kaviraj Darren	11th April (3 pm)	In Progress

14.0 References

Deskera. (2022). *Top 10 reasons why restaurants fail (and how to avoid it)*. Retrieved from https://www.deskera.com/blog/restaurant-challenges-solutions/

Foodpanda. (n.d.). *Foodpanda for Business.* Retrieved April 12, 2025, from https://www.foodpanda.my/corporate/sign-up/enterprise

Gojek. (n.d.). *Gojek for Business.* Retrieved April 12, 2025, from https://www.gojek.com/en-id/gocorp

Google Cloud. (2023). *AI and ML in small business retail*. Retrieved from https://cloud.google.com/solutions/retail

GrabX - *Grab's first-ever product event* | Grab SG. (2025). Grab. https://www.grab.com/sg/grabx/

Marks, G. (2016, October 14). *Almost half of retail grocery store managers are just guessing about their inventory*. The Washington Post. https://www.washingtonpost.com/news/on-small-business/wp/2016/10/14/almost-half-of-retail-grocery-store-managers-are-just-guessing-about-their-inventory/?utm_source=chatgpt.com

McKinsey & Company. (2021). *The value of getting personalization right—or wrong—is multiplying*. Retrieved from https://www.mckinsey.com/business-functions/growth-marketing-and-sales/our-insights/the-value-of-getting-personalization-right-or-wrong-is-multiplying

Salesforce. (2023). *What is CRM?* Retrieved from https://www.salesforce.com/crm/what-is-crm/

Swiggy. (n.d.). Swiggy Corporate & Merchant Services. Retrieved April 12, 2025, from https://www.swiggy.com/corporate/our-business/

The Star. (2022, September 27). *Malaysia generates 17,000 tonnes of food waste daily, says the Environment Minister*. Retrieved from https://www.thestar.com.my/news/education/2024/02/04/when-good-food-goes-to-waste

WhatsApp Business. (2023). *Automate and personalize customer interactions*. Retrieved from https://business.whatsapp.com/

World Bank. (2021). *The role of digital tools in supporting MSMEs in Southeast Asia*. Retrieved from Xhttps://documents1.worldbank.org/curated/en/099515009292224182/pdf/P1760 8901a9db608909f5b02980d48c4e28.pdf

Zendesk. (2022). *What is CRM software?* Retrieved from https://www.zendesk.com/sell/crm/what-is-crm/