RSET
RAJAGIRI SCHOOL OF
ENGINEERING & TECHNOLOGY
(AUTONOMOUS)

Project Report/Seminar report On

Title

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Name of the Programme

By

Name (UID)

Under the guidance of

Name of the Guide

Name of the Department
Rajagiri School of Engineering & Technology (Autonomous)
(Parent University: APJ Abdul Kalam Technological University)
Rajagiri Valley, Kakkanad, Kochi, 682039
July 2023

CERTIFICATE

This is to certify that the project report/seminar report entitled ***"Title"*** *is a bonafide record of the work done by* ***Student Name (UID)****, submitted to the Rajagiri School of Engineering & Technology (RSET) (Autonomous) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in "Name of the Programme" during the academic year 20XX-20XX.*

Project Guide	Project Co-guide	Project Co-ordinator
Designation	Designation	Designation
Dept. Name	Dept. Name	Dept. Name
RSET	RSET	RSET

Name of HoD
Designation
Dept. Name
RSET

ACKNOWLEDGMENT

I wish to express my sincere gratitude towards **Name**, Principal of RSET, and "Name of HoD", Head of the Department of "Name of the Department" for providing me with the opportunity to undertake my project, "Project Title".

I am highly indebted to my project coordinators, **Name(s)**, Designation, Department, for their valuable support.

It is indeed my pleasure and a moment of satisfaction for me to express my sincere gratitude to my project guide **Name of Guide** for his/her patience and all the priceless advice and wisdom he/she has shared with me. I also express my sincere thanks to my co-guide(s), **Name of Co-Guide** for his/her support. *(Edit the contents accordingly)*

Last but not the least, I would like to express my sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

Name of Student

Abstract

Insert your abstract here. The abstract should include a concise and clear description of the project work done. It should highlight the advantages of the project compared to existing works.

Contents

List of Abbreviations

Acronym - Expansion

List of Figures

List of Tables

Chapter 1

Introduction

Chapter introduction goes here.

1.1 Background

This section should outline the background of the project under consideration highlighting current scenarios and its importance. Maximum content is around 1 page [1].

1.2 Problem Definition

This section should mention the aim of the project. The problem should be defined in one or two sentences.

Figure 1.1: Insert your images here, and provide necessary captions.

1.3 Scope and Motivation

This section mentions scope and motivation, which should be written as two paragraphs. The first paragraph describes the scope, whereas the second one describes the motivation. Write in around 5 sentences each.

You may insert tables into your document using the given code:

Table 1.1: Insert table caption here

Title 1	**Title 2**	**Title 3**
1	Content 1	Content 2
2	Content 3	Content 4

1.4 Objectives

- This section should be a numbered list. Five to six objectives are encouraged.

1.5 Challenges

This section briefs the challenges involved in the project in two or three sentences.

1.6 Assumptions

This section briefs the assumptions in the project in two or three sentences or as a numbered list.

1.7 Societal / Industrial Relevance

This section describes where the project can be applied, either for the society or the industry. Write the relevance applicable for the work.

1.8 Organization of the Report

This section should outline a roadmap of the contents in the report.

Chapter conclusion goes here.

Chapter 2

Literature Survey

Chapter introduction goes here.

2.1 Section 1 Heading

Contents [2]

(a) First subfigure.

(b) Second subfigure.

(c) Third subfigure.

Figure 2.1: Creating subfigures.

2.2 Section 2 Heading

Contents

2.2.1 Subsection Heading

Contents

2.3 Summary and Gaps Identified

This is the most important section of Chapter 2. This subsection has two parts (i) summary and (ii) gaps identified. Summary can be a tabular form mentioning the advantages/disadvantages associated with each title. The gaps identified can be a numbered list of around four or five points mentioning what is lacking in the current state of art.

Chapter conclusion goes here. Here's a structured chapter in the form of **System Architecture** with sections and subsections corresponding to each component you've provided:

—

Chapter 3

System Architecture

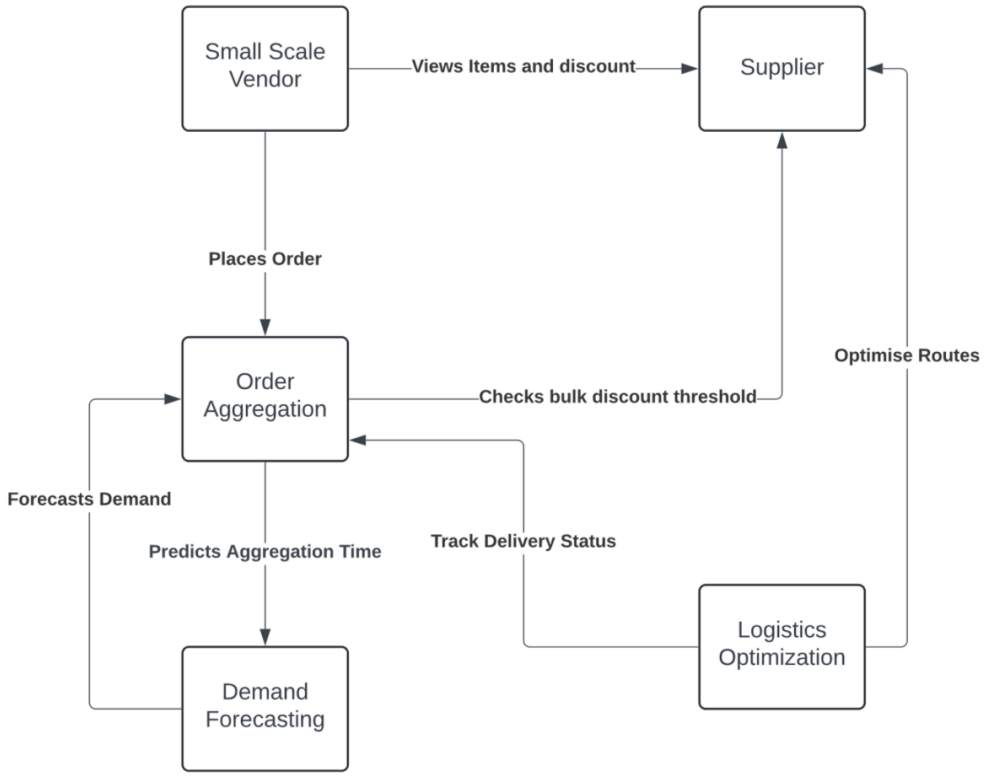

Figure 3.1: System Architecture of Vendor Collaboration Platform

Figure 3.1 illustrates the system architecture for a Vendor Collaboration Platform, designed to support small-scale vendors by facilitating bulk order discounts, demand forecasting, and logistics optimization. This architecture provides a clear overview of the platform's components and how they work together to streamline the entire process from order initiation to optimized delivery. Here's a breakdown of the key aspects and importance of this architecture:

1. **High-Level Overview:** The architecture serves as a roadmap, showing the main

stages of the platform's workflow. From initial product viewing and order placement by vendors, through demand forecasting, order aggregation, and logistics optimization, this high-level overview provides all stakeholders—including vendors, suppliers, and platform administrators—with a clear understanding of the system's core functionality.

2. **Clarifies Data Flow:** The architecture clarifies the flow of data within the system, beginning with the acquisition of order and sales data. This data undergoes preprocessing and feature extraction, followed by demand forecasting with the Neural Prophet model. The forecast data then flows into the Order Aggregation module, which combines vendor orders to meet bulk purchase thresholds. Finally, optimized delivery routes are created through the Logistics Optimization module, using Green Routing techniques. This sequential data flow enhances system transparency and makes it easy to identify potential bottlenecks.

3. **Enables Collaborative Order Aggregation:** The architecture highlights how order aggregation enables small-scale vendors to achieve bulk discounts. By aggregating orders from multiple vendors, the platform ensures minimum order quantities (MOQs) are met, enabling vendors to access bulk discounts typically reserved for larger chains. This aspect of the system is crucial for enhancing the competitiveness of small vendors.

4. **Supports Logistics Optimization through Green Routing:** The architecture emphasizes the logistics optimization approach used in the platform. The Green Routing component incorporates path flexibility and service time windows to create efficient and environmentally friendly delivery routes. This optimization minimizes transportation costs and enhances delivery reliability for vendors, contributing to a streamlined and cost-effective supply chain.

5. **Enables Real-Time Notifications and Updates:** The system architecture includes a notification mechanism that provides real-time updates to vendors on order status, inventory levels, and logistics. These notifications foster transparency, enabling vendors to make informed decisions about inventory and delivery schedules.

6. **Facilitates Collaboration and Communication Among Stakeholders:** The

system architecture acts as a shared framework, providing a common language for platform users, developers, and suppliers. It allows stakeholders to better understand the interactions between components and collaborate effectively toward improving platform functionality. This collaborative framework is essential for ensuring smooth operations and adapting to vendor needs.

Overall, the system architecture is crucial for understanding the Vendor Collaboration Platform. It provides a clear visual representation of the platform's workflow, highlighting key features such as order aggregation, demand forecasting, logistics optimization, and real-time notifications. These components work together to enhance vendor competitiveness by offering cost-saving opportunities, efficient delivery solutions, and improved collaboration with suppliers.

3.1 Dataset

Purpose: This dataset supports research in sales forecasting, specifically at the item and store levels. The objective is to predict the next three months of sales for individual items at various store locations, applicable for demand forecasting, inventory management, and sales optimization.

Data Collection: This dataset was collected from multiple store locations, representing real-world sales data with variations in store performance and item demand. It provides diverse daily sales records for individual items across different stores, enabling demand pattern analysis.

Data Fields:

date: Date of each sale record. There are no adjustments for holidays or store closures, simplifying the analysis but possibly requiring external data for event-based effects.

store: A unique identifier for each store, allowing for store-level analysis and capturing unique characteristics and item demand differences.

item: A unique identifier for each item sold, enabling item-level demand forecasting and cross-store analysis.

sales: The number of items sold at a particular store on a given date, which is the target variable for forecasting models.

File Descriptions:

train.csv: Contains historical sales data used for training forecasting models.

test.csv: Provides the test data where future sales predictions are needed, with a time-based split for realistic scenario testing.

sample_submission.csv: A sample submission file showing the required format for submitting sales forecasts.

Usage in Research: This dataset supports the development and validation of time series forecasting models. It allows researchers to explore seasonality, trends, and external influences on sales. Models like Neural Prophet or ARIMA can be applied, along with machine learning techniques for enhanced forecasting accuracy.

Challenges and Opportunities: Although the dataset lacks explicit holiday or store event information, researchers may incorporate external data to improve forecasts. This provides an opportunity to explore techniques that account for seasonality and trend modeling, such as Fourier series for seasonality or external regressors.

Access: This dataset is easily accessible, enabling collaborative studies on sales forecasting and providing a resource for testing and improving demand forecasting algorithms.

3.2 Preprocessing

Data preprocessing is essential in preparing the dataset for accurate prediction. This involves handling missing values, scaling numerical data, creating time-based features, and engineering lagged variables. Each step in preprocessing enhances the model's performance by structuring meaningful inputs.

Handling Missing Values

In sales datasets, missing values often occur due to issues in data collection. Missing values are handled using various methods:

- **Forward Fill**: Uses the last known value to fill missing entries.
- **Backward Fill**: Uses the next known value to fill missing entries.
- **Interpolation**: Estimates missing values based on surrounding data points.

Scaling and Normalization

To ensure that features are on the same scale, Min-Max scaling is applied to rescale values between 0 and 1.

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{3.1}$$

where X is the original value, $X_{\min}$ is the minimum value, and $X_{\max}$ is the maximum value. This scaling process helps models like Neural Prophet to converge more efficiently.

Table 3.1: Example of Min-Max Scaling

Original Sales	**Min Value**	**Max Value**	**Scaled Sales**
50	0	100	0.5
30	0	100	0.3

Time-Based Feature Engineering

Time-based features capture seasonal patterns and trends that are crucial for forecasting. Extracted features include:

- **Day of the Week**: Helps capture day-specific sales trends.
- **Week of the Year**: Highlights seasonal variations.
- **Month**: Reflects monthly demand patterns.

Lagged Sales Features

Lagged variables incorporate past sales data, helping the model to learn patterns over time.

Lagged Sales Formula

$$\text{Lag}_{t-k} = \text{Sales}_{t-k} \tag{3.2}$$

where k is the lag period, e.g., 7 days for a weekly lag.

Rolling Mean Sales Formula

$$\text{Rolling Mean}_t = \frac{1}{N} \sum_{i=t-N+1}^{t} \text{Sales}_i \tag{3.3}$$

where N is the window size, such as 30 days.

Rolling Standard Deviation Formula

$$\text{Rolling Std Dev}_t = \sqrt{\frac{1}{N} \sum_{i=t-N+1}^{t} (\text{Sales}_i - \text{Rolling Mean}_t)^2} \tag{3.4}$$

These rolling features smooth fluctuations and highlight variability in sales.

Table 3.2: Examples of Lagged and Rolling Features

Date	**Sales**	**7-Day Rolling Mean**
2024-11-01	150	140
2024-11-02	160	145

Encoding Categorical Variables

Store and item IDs are encoded as numerical values. One-hot encoding is applied to create binary variables for each unique category.

Table 3.3: Example of One-Hot Encoding

Store ID	**Store_1**	**Store_2**	**Store_3**
1	1	0	0
2	0	1	0

Conclusion

This preprocessing approach ensures the data is clean, consistent, and enriched with meaningful features, laying a foundation for effective demand forecasting. The structured data allows the model to capture trends and seasonal effects, improving forecast accuracy.

3.3 Feature Extraction

Feature extraction is a crucial step in analyzing and understanding data for demand forecasting, especially in a retail context. By extracting relevant features, we can uncover patterns in sales data that support more accurate predictions. This process aims to capture the key temporal characteristics in sales trends and seasonality, which play a critical role in demand planning and inventory management.

Importance of Feature Extraction:

- **Dimensionality Reduction**: Raw sales data, when augmented with time-based features, enables the model to focus on critical patterns without unnecessary complexity, reducing dimensionality and improving model performance.
- **Relevance to Demand Patterns**: Relevant features are those that capture seasonal trends and fluctuations in demand. For example, features representing daily, weekly, and monthly patterns allow the model to recognize recurring trends.
- **Improved Model Performance**: Extracting informative features enables models to generalize better to unseen data, ultimately increasing the effectiveness of demand forecasting and inventory optimization.

3.3.1 Time-Based Features

Time-based features capture the cyclical nature of retail sales, such as daily, weekly, and monthly patterns, which are essential for forecasting future demand accurately.

- **Day of the Week**: This feature captures weekly sales trends and highlights days with higher or lower demand, useful for identifying daily sales patterns that may recur each week.
- **Week of the Year**: A feature representing the week in the year, useful for capturing demand variations at different times of the year, especially if certain weeks have increased activity due to events or holidays.
- **Month**: Monthly features are vital for capturing seasonal trends, especially in cases where sales volumes vary significantly by month (e.g., holiday seasons or back-to-school months).

- **Quarter**: Dividing the year into quarters can capture broader seasonal effects and assist in longer-term planning and budgeting.

Adding these time-based features enriches the data with cyclical information, helping models capture patterns that align with specific times of the year.

3.3.2 Lagged Variables

Lagged variables refer to features created by using past values of the target variable (sales) to inform current predictions. Lagged features provide context on recent sales history and are particularly useful for time series models that depend on trends and seasonality.

- **Previous Day Sales**: This feature captures the sales level from the prior day, offering insights into day-to-day changes in demand, valuable for identifying short-term fluctuations.
- **Weekly Lagged Sales (e.g., 7-Day Lag)**: A 7-day lag captures the sales from the same day in the previous week, making it useful for capturing weekly patterns and seasonality.
- **Monthly Lagged Sales (e.g., 30-Day Lag)**: Using a 30-day lag, or a similar monthly lag, allows the model to consider monthly sales cycles and identify monthly sales trends.

By incorporating these lagged variables, the model is able to learn from recent sales patterns, improving its ability to detect trends and anticipate changes in demand.

Lagged Sales Formula

$$\text{Lag}_{t-k} = \text{Sales}_{t-k} \tag{3.5}$$

where k is the lag period, e.g., 7 days for a weekly lag.

Rolling Mean Sales Formula

$$\text{Rolling Mean}_t = \frac{1}{N} \sum_{i=t-N+1}^{t} \text{Sales}_i \tag{3.6}$$

where N is the window size, such as 30 days.

Rolling Standard Deviation Formula

$$\text{Rolling Std Dev}_t = \sqrt{\frac{1}{N} \sum_{i=t-N+1}^{t} (\text{Sales}_i - \text{Rolling Mean}_t)^2} \tag{3.7}$$

These rolling features smooth fluctuations and highlight variability in sales.

Table 3.4: Examples of Lagged and Rolling Features

Date	**Sales**	**7-Day Rolling Mean**
2024-11-01	150	140
2024-11-02	160	145

3.4 Demand Forecasting with Neural Prophet

Demand forecasting is a fundamental component of this platform, enabling vendors to make data-driven decisions about inventory management and order planning. For this project, we utilize the Neural Prophet model, an advanced time-series forecasting tool designed to handle complex seasonality, trend, and holiday effects, making it particularly suited for retail sales forecasting.

Neural Prophet builds on the principles of Facebook's Prophet model but incorporates neural network architecture to improve predictive performance, especially when seasonal patterns and external factors significantly impact demand. By leveraging Neural Prophet, this platform provides vendors with accurate and timely insights into anticipated demand, helping them optimize inventory and align procurement strategies.

3.4.1 Model Components and Workflow

Neural Prophet decomposes sales data into key components—trend, seasonality, holidays, and regressors—that together account for the variations observed in demand. This decomposition enables the model to capture both recurring patterns and unique spikes associated with specific time intervals. The following components are configured in Neural Prophet to suit our forecasting requirements:

- **Trend Component**: A piecewise linear trend with configurable changepoints enables the model to adapt to both gradual and sudden shifts in demand, capturing

long-term growth or decline.

- **Seasonality**: Modeled using Fourier series, Neural Prophet supports daily, weekly, and yearly seasonality. For our retail platform, capturing weekly and monthly patterns is crucial for understanding cycles influenced by factors like weekends and holidays.

- **Holiday Effects**: Neural Prophet allows for the inclusion of known holiday effects, capturing demand spikes during these periods through the use of external regressors.

- **Lagged Variables and Regressors**: Additional regressors, such as lagged sales data, are incorporated to account for recent demand patterns, improving forecast accuracy.

3.4.2 Model Implementation Process

The implementation of Neural Prophet for demand forecasting follows these steps:

1. **Data Preparation**: Historical sales data is preprocessed and augmented with time-based features, such as day of the week and month, to capture demand cycles. Missing values are handled, and data is scaled for model efficiency.

2. **Feature Engineering**: Lagged variables are created to include recent sales trends. Holiday effects and custom seasonalities are also specified to incorporate relevant demand spikes.

3. **Model Training**: The preprocessed data, including lagged features and regressors, is input into Neural Prophet, which is then trained on past sales data. Hyperparameters, like the number of Fourier terms for seasonality, are optimized to achieve high accuracy.

4. **Forecast Generation**: The trained model generates forecasts for the specified time horizon, providing sales predictions for each store-item combination. These forecasts enable vendors to anticipate demand fluctuations.

3.4.3 Benefits for Demand Forecasting

Using Neural Prophet for demand forecasting provides several key benefits:

- **Accuracy**: By decomposing sales data into trend, seasonality, and holiday effects, Neural Prophet achieves high accuracy. Incorporating lagged variables and external regressors further improves performance.
- **Adaptability**: Neural Prophet's flexible architecture ensures accurate forecasts even as demand patterns shift over time.
- **Scalability**: Neural Prophet supports high-frequency data and large datasets, making it well-suited for our platform as it scales.

Table 3.5: Example of Fourier Series Representation for Seasonality

Fourier Term	**Weekly Seasonality**	**Monthly Seasonality**
$\sin\left(\frac{2\pi t}{7}\right)$	0.35	-0.47
$\cos\left(\frac{2\pi t}{7}\right)$	0.67	0.52
$\sin\left(\frac{2\pi t}{30}\right)$	-0.14	0.88

This Fourier series representation allows Neural Prophet to effectively model seasonality by capturing cyclical patterns with both weekly and monthly components, enhancing the model's adaptability to demand cycles.

3.5 Order Aggregation using Genetic Algorithm

Order aggregation is a crucial optimization process in this platform, designed to help small-scale vendors combine orders to meet minimum order quantities (MOQs) set by suppliers and leverage cost savings through bulk purchasing. The Genetic Algorithm (GA) is employed as the optimization technique for order aggregation due to its efficiency in exploring vast solution spaces and its adaptability to constraints like MOQ. By enabling vendors to combine orders, the platform fosters collaboration, reduces costs, and improves supply chain efficiency.

3.5.1 Genetic Algorithm for Order Aggregation

The Genetic Algorithm is an evolutionary optimization technique that mimics the principles of natural selection. For the order aggregation problem, it searches for the optimal way to combine orders from multiple vendors to achieve the best cost savings while ensuring each aggregated order meets the MOQ. The algorithm proceeds through several steps:

1. **Population Initialization**: An initial population of possible order combinations is generated, with each individual representing a unique aggregation of orders from multiple vendors. Each solution or individual encodes which vendor orders are combined and the resulting total quantity.

2. **Fitness Evaluation**: The fitness function assesses each individual's ability to meet MOQs and maximize cost savings. Solutions that meet MOQs and achieve bulk purchasing discounts receive higher fitness scores.

3. **Selection**: Based on fitness, individuals are selected to act as "parents" for the next generation, favoring solutions that better meet the cost-saving objectives and MOQ constraints.

4. **Crossover**: Pairs of selected individuals undergo crossover to produce new "offspring" solutions. This exchange of order combinations allows for the exploration of new aggregation strategies by mixing elements of high-performing solutions.

5. **Mutation**: To introduce diversity, mutation randomly alters parts of some solutions, creating new order combinations that may explore previously unexplored parts of the solution space.

6. **Termination**: The algorithm iterates through generations until it converges on an optimal solution or meets a pre-set termination condition, such as a maximum number of generations or minimal improvement in fitness.

3.5.2 Optimization Constraints and Objectives

The GA is configured with specific objectives and constraints tailored to the order aggregation needs of the platform:

- **Minimum Order Quantity (MOQ)**: Each combined order must satisfy the MOQ constraint to be eligible for supplier discounts. Solutions failing to meet MOQ are penalized in fitness scoring.
- **Cost Minimization**: The primary objective of the GA is to minimize the cost per unit for vendors by aggregating orders to achieve bulk discounts. Each fitness evaluation calculates the potential cost savings per solution, favoring combinations with the highest savings.
- **Order Compatibility**: Certain products or order combinations may be subject to compatibility constraints, ensuring only compatible products are aggregated.

3.5.3 Benefits of Genetic Algorithm-Based Order Aggregation

Using the Genetic Algorithm for order aggregation brings several advantages to the platform:

- **Scalability**: GA can efficiently process large numbers of vendors and orders, making it suitable for a platform that serves a dynamic vendor base.
- **Adaptability**: By accommodating various constraints, such as MOQ and product compatibility, the GA provides an adaptable solution for different order aggregation scenarios.
- **Cost Efficiency**: By maximizing the probability of achieving bulk purchasing discounts, the GA supports vendors in minimizing unit costs, directly benefiting their profit margins.

Table 3.6: Example of Genetic Algorithm Parameters for Order Aggregation

Parameter	**Value**
Population Size	100
Crossover Rate	0.8
Mutation Rate	0.1
Number of Generations	50

These parameters are tuned to balance the exploration of possible order combinations and the convergence to an optimal solution, ensuring that the GA efficiently identifies high-quality order aggregation strategies.

3.6 Logistics Optimization with Green Routing

The logistics optimization module in this system employs Green Routing techniques to minimize transportation costs and reduce environmental impact. This module is essential for supporting sustainable logistics practices while ensuring timely and cost-efficient delivery for vendors. The Green Routing optimization consists of two primary components: Path Flexibility and Service Time Window, which together allow the system to create optimal delivery routes that are both efficient and environmentally conscious.

3.6.1 Path Flexibility

Path Flexibility enables the routing algorithm to identify and select routes that minimize fuel consumption and emissions while also balancing cost efficiency. By incorporating alternative paths, the system can avoid congested areas, reduce travel time, and decrease carbon emissions. This component of Green Routing helps optimize logistics in the following ways:

- **Emission Reduction**: The system prioritizes routes with lower fuel consumption, thereby reducing CO_2 emissions.
- **Dynamic Traffic Adaptation**: Real-time traffic data is utilized to identify less congested routes, which can improve delivery time and reduce idle fuel consumption.
- **Cost-Efficient Routing**: By choosing paths that reduce fuel use and travel distance, the system lowers transportation costs, benefiting both vendors and the environment.

$$\text{Fuel Cost} = \text{Distance} \times \text{Fuel Consumption Rate} \times \text{Fuel Price} \tag{3.8}$$

where Distance is the total travel distance for a route, Fuel Consumption Rate represents fuel usage per unit distance, and Fuel Price is the current price of fuel. Minimizing

this cost function through Path Flexibility allows the system to prioritize routes that save both fuel and expenses.

3.6.2 Service Time Window

Service Time Window refers to the allocated time frames for deliveries at each vendor location. By adhering to designated service windows, the system optimizes scheduling to ensure that all deliveries occur within specific time constraints, enhancing reliability and customer satisfaction. This component is beneficial for logistics in multiple ways:

- **Optimized Delivery Schedule**: Ensuring deliveries are completed within set time windows reduces waiting times and helps vendors plan their inventory and sales operations more effectively.
- **Resource Efficiency**: By aligning delivery routes with service windows, the system minimizes the likelihood of delays or failed deliveries, improving resource utilization.
- **Customer Satisfaction**: Timely deliveries within specified windows improve vendor satisfaction and service reliability, enhancing overall platform reputation.

$$\text{Penalty Cost} = \begin{cases} 0, & \text{if Arrival Time} \leq \text{Service End Time} \\ k \times (\text{Arrival Time} - \text{Service End Time}), & \text{if late} \end{cases} \tag{3.9}$$

where k is the penalty coefficient for late deliveries. This equation ensures that the optimization process considers any potential penalties for exceeding the designated time window, thereby prioritizing timely deliveries.

3.6.3 Route Optimization Process

The Green Routing optimization process combines Path Flexibility and Service Time Window to create the most efficient routes under given constraints. The process operates as follows:

1. **Route Generation**: An initial set of possible routes is generated, each considering various factors such as distance, fuel consumption, and service windows.

2. **Cost Evaluation**: Each route is evaluated based on its fuel cost, emission impact, and penalty cost for potential delays. Routes with lower total costs are prioritized.

3. **Optimization**: Using algorithms that consider both path flexibility and time window constraints, the system iterates through routes to refine and select the optimal path for each delivery cycle.

4. **Real-Time Adjustments**: As conditions change (e.g., traffic congestion), the system can dynamically adjust the selected route to maintain cost efficiency and minimize delays.

Table 3.7: Example of Route Evaluation Criteria

Route	**Fuel Cost ($)**	**CO_2 Emissions (kg)**	**Penalty Cost ($)**
Route A	150	120	0
Route B	135	115	15
Route C	140	110	10

The Logistics Optimization module's Green Routing approach is integral to reducing environmental impact and operational costs in the platform. By combining Path Flexibility and Service Time Window, the system generates efficient, sustainable delivery routes that align with both vendor expectations and eco-friendly practices. This dual-component approach ensures that the platform remains competitive by providing vendors with reliable, timely, and cost-effective logistics solutions.

3.7 Model Evaluation and Optimization

Model Evaluation and Optimization are critical steps in ensuring that the demand forecasting model achieves high accuracy and generalizes well to new data. This section details the three main components of this process: Evaluation, Hyperparameter Tuning, and Iterative Optimization. Together, these steps help enhance model performance by systematically assessing accuracy, refining model parameters, and iteratively improving results.

3.7.1 Evaluation

Evaluation involves assessing the model's accuracy and performance on a validation dataset. This step is essential to verify that the model's predictions align with actual demand patterns and to identify potential areas for improvement. Key evaluation metrics used in this project include Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), which are calculated as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.10}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.11}$$

where y_i represents the actual values, $\hat{y}_i$ represents the predicted values, and n is the number of predictions. Lower values of MAE and RMSE indicate better model accuracy and less deviation between predicted and actual values.

Table 3.8: Evaluation Metrics for Forecasting Model

Model	**MAE**	**RMSE**
Baseline Model	12.5	15.3
Optimized Model	10.2	12.8

These metrics provide a clear understanding of the model's current accuracy, allowing us to assess the impact of further tuning and optimization.

3.7.2 Hyperparameter Tuning

Hyperparameter tuning is the process of selecting the best model parameters to improve forecast accuracy. In this project, techniques such as Grid Search were employed to explore a range of hyperparameter combinations, with the objective of finding the optimal configuration for the model.

- **Grid Search**: This technique systematically tests predefined ranges of hyperparameters, such as learning rate, seasonal periods, and growth components, to identify the best-performing combination.

- **Cross-Validation**: Each set of hyperparameters is evaluated through cross-validation, ensuring that the tuning process does not lead to overfitting on the training data.

The result of hyperparameter tuning is an optimized model with improved predictive power, as it balances complexity and accuracy.

Table 3.9: Hyperparameter Tuning Results

Parameter	**Range Tested**	**Optimal Value**
Learning Rate	0.001 - 0.1	0.01
Seasonal Period	7, 14, 30	14
Growth Rate	Linear, Logistic	Logistic

This table summarizes the hyperparameters tuned and the optimal values identified, which contribute to improved forecast accuracy.

3.7.3 Iterative Optimization

The Iterative Optimization process enables continuous refinement of the model by re-evaluating and adjusting parameters based on performance metrics. This approach ensures that the model adapts to changes in data patterns and maximizes predictive accuracy over time. The steps involved in iterative optimization include:

1. **Feature Engineering Refinement**: Based on evaluation results, new features may be engineered or existing features modified to capture additional seasonal or trend components.

2. **Parameter Adjustment**: Hyperparameters may be re-tuned or adjusted as new data insights emerge, improving model generalization.

3. **Model Re-training**: The model is periodically re-trained with the updated parameters and features, which allows for adaptation to recent demand trends.

$$\text{Improvement Factor} = \frac{\text{Old RMSE} - \text{New RMSE}}{\text{Old RMSE}} \tag{3.12}$$

This formula calculates the improvement factor in RMSE as an indicator of optimization progress, helping to assess the effectiveness of each iteration.

Table 3.10: Iterative Optimization Performance Over Time

Iteration	**MAE**	**RMSE**
Initial Model	12.5	15.3
After 1st Optimization	11.2	13.9
After 2nd Optimization	10.2	12.8

The Evaluation and Optimization processes ensure that the forecasting model is continually enhanced to meet accuracy and reliability requirements. By implementing evaluation metrics, rigorous hyperparameter tuning, and iterative optimization, this project achieves a robust demand forecasting model capable of adapting to evolving data patterns and improving performance over time.

3.8 System Integration and Notifications

The System Integration and Notifications module unifies all components in the platform, ensuring seamless data flow and robust communication across users. This module facilitates the synchronization of forecasting, order aggregation, and logistics, enabling vendors to make informed decisions and manage operations efficiently. The system's architecture has been designed to optimize user interaction and data processing, allowing for real-time updates and notifications.

3.8.1 System Integration

System integration connects each component in the platform, enabling a smooth exchange of information necessary for coordinated operations. This integration links the demand forecasting, order aggregation, and logistics optimization modules, ensuring that all components work in harmony. Key integration processes include:

- **Data Pipeline Management**: The platform manages a centralized data pipeline, which allows information such as sales forecasts, order details, and logistics schedules to flow seamlessly. Data from the forecasting model directly informs order aggregation, which in turn updates logistics routing.

- **Real-Time Inventory and Order Status Updates**: Inventory levels and order statuses are dynamically updated as transactions occur, keeping all users informed of current stock and order progress. When vendors place an order, the system automatically checks available stock, order quantities, and logistics parameters before initiating further processes.

- **Cross-Module Communication**: Communication protocols enable real-time interaction among components, with APIs connecting external data sources or integrating third-party logistics providers if needed. This setup enables efficient responses to fluctuations in demand or changes in supply chain schedules.

3.8.2 Notifications

The Notifications feature is integral to user engagement, keeping vendors updated on critical aspects of order processing, inventory status, and delivery schedules. Notifications are triggered by specific events within the platform, ensuring timely and relevant information delivery to users. The key notifications implemented in this system include:

- **Order Status Updates**: Vendors receive updates on order processing, confirmation of joint orders, and shipment tracking. This transparency allows vendors to plan their inventory management and sales strategy effectively.

- **Inventory Level Alerts**: Low inventory alerts notify vendors when stock reaches a specified threshold, ensuring they can replenish products before stockouts occur. High-demand items are monitored, and vendors are alerted to prevent missed sales opportunities.

- **Delivery Schedule Notifications**: Notifications include estimated delivery times and routing updates, assisting vendors in planning for incoming shipments and scheduling resources for unloading or further distribution.

Notifications are configured to reach users through multiple channels, such as in-app alerts, emails, and SMS, allowing vendors to remain updated even when away from the platform.

Table 3.11: Notification Types and Triggers

Notification Type	**Trigger Event**	**Delivery Channel**
Order Status Update	Order Confirmation, Shipment Dispatch	In-App, Email
Inventory Level Alert	Low Stock Threshold Reached	In-App, SMS
Delivery Schedule	Route Finalized, Estimated Delivery Time	In-App, Email

This chapter has outlined the system's architecture and described each component's role in creating a cohesive platform for demand forecasting, order aggregation, and logistics management. Through effective system integration and real-time notifications, the platform provides a unified experience that supports vendors in optimizing their operations, managing inventory, and coordinating logistics. The collaborative approach facilitated by this platform empowers vendors to maintain competitive advantages, meeting demand efficiently and enhancing overall supply chain resilience.

Chapter 4

DESIGN AND MODELING

Design and modeling are essential phases in the software development lifecycle, playing a crucial role in the creation of effective and wellstructured systems. These phases involve the conceptualization, planning, and representation of a software solution before its actual implementation.

4.1 Activity Diagram

An activity diagram is a type of UML (Unified Modeling Language) diagram that visually represents the flow of activities in a system or business process. It is commonly used in software development to illustrate the dynamic aspects of a system and describe how different activities or tasks interact with each other. Activity diagrams are particularly useful for modeling the workflow within a system and for understanding the sequence of actions that take place.

4.1.1 Key elements of Activity Diagrams

- **Activity:** Represented by rounded rectangles, activities are the specific tasks or actions that are performed within the system. These can range from simple operations to complex processes.
- **Transitions:** Arrows connecting activities indicate the flow or transition from one activity to another. The direction of the arrow shows the order of execution.
- **Decision Nodes:** Diamonds are used to represent decision points in the workflow. Depending on certain conditions, the process may take different paths.
- **Fork and Join Nodes:** Fork nodes (split) and join nodes (merge) show parallel or concurrent activities. Forks represent the divergence of multiple flows, while joins

represent their convergence.

- **Initial and Final Nodes:** Circles are used to denote the start (initial node) and end (final node) of the activity diagram. The initial node represents the beginning of the process, and the final node indicates the conclusion.

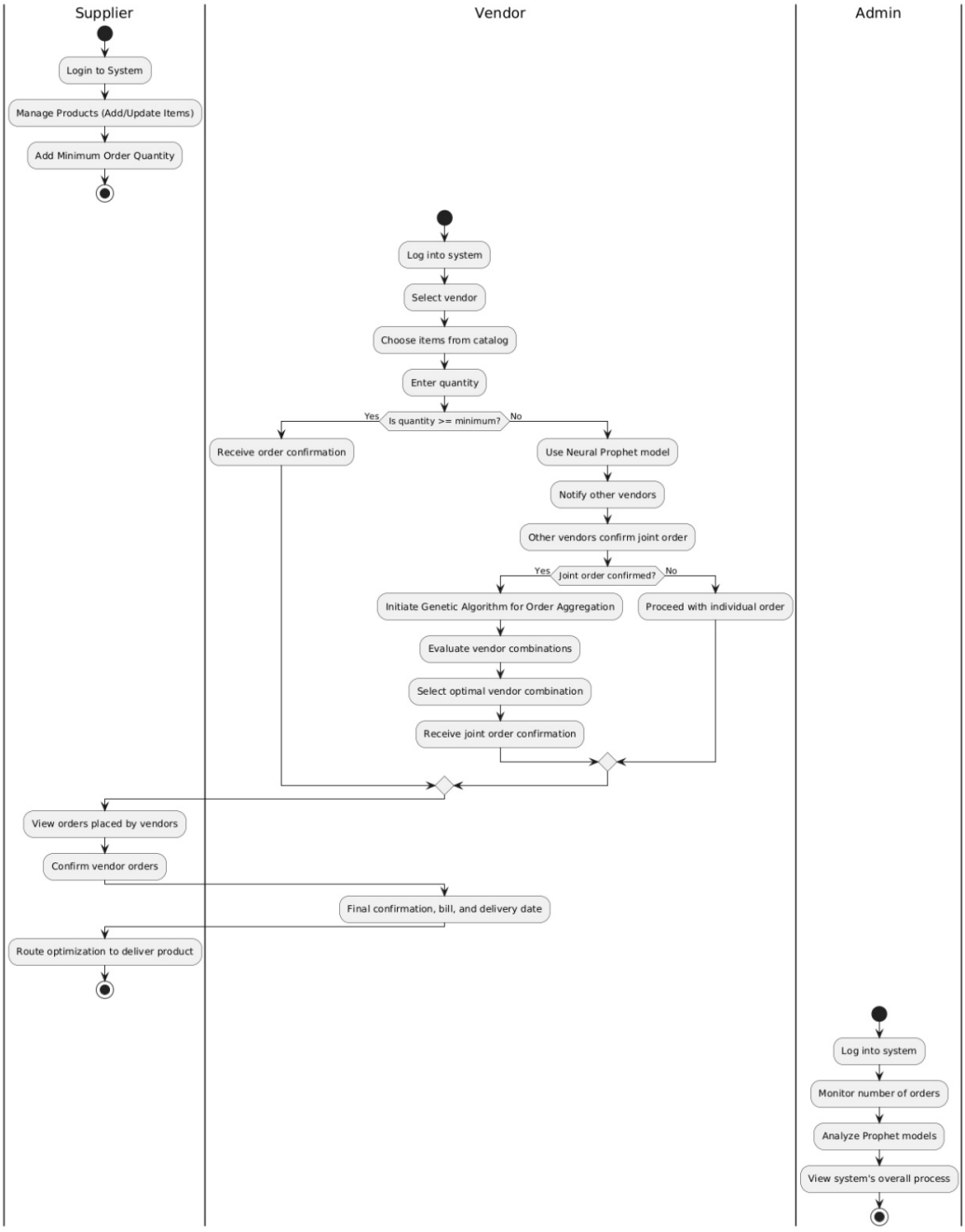

Figure 4.1: Activity Diagram

4.1.2 Key Entities and Components

1. **Supplier Activities**

   (a) **Login to System:** The supplier logs into the system to access functionalities for managing products and tracking orders.

   (b) **Manage Products (Add/Update Items):** Suppliers add or update product listings to ensure the catalog is up-to-date and accurate.

   (c) **Add Minimum Order Quantity:** Suppliers set a minimum order quantity for each product, preventing inefficient processing of very small orders.

   (d) **View Orders Placed by Vendors:** Suppliers can view pending orders from vendors, preparing them for order fulfillment.

   (e) **Confirm Vendor Orders:** Suppliers review and confirm vendor orders, moving them to the fulfillment phase.

   (f) **Route Optimization for Delivery:** The supplier optimizes delivery routes to reduce logistics costs and ensure timely product delivery.

2. **Vendor Activities**

   (a) **Log into System:** Vendors start by logging into the system to access the product catalog and ordering functionalities.

   (b) **Select Vendor and Choose Items from Catalog:** Vendors select suppliers to order from, browsing the catalog to choose items.

   (c) **Enter Quantity:** Vendors input desired quantities for each item, triggering a check against the minimum order quantity.

   (d) **Quantity Check (Yes/No):**

   - **If Quantity Meets Minimum Requirement:** The order is confirmed directly.
   - **If Quantity Is Below Minimum Requirement:**
     i. **Use Neural Prophet Model:** The system uses a Neural Prophet model to forecast demand and determine if combining orders with other vendors can reach the minimum order threshold.

ii. **Notify Other Vendors:** Other vendors are notified to consider joining a joint order.

iii. **Joint Order Confirmation Check (Yes/No):**

- **If Other Vendors Agree:** The joint order is confirmed.
- **If Not:** The vendor proceeds with an individual order.

iv. **Initiate Genetic Algorithm for Order Aggregation:** The system uses a genetic algorithm to identify the best vendor combinations for the joint order.

v. **Evaluate Vendor Combinations:** Different combinations are assessed to find an optimal one that meets requirements.

vi. **Select Optimal Vendor Combination:** The system selects the optimal vendor combination, and a joint order confirmation is received.

vii. **Receive Final Confirmation:** Vendors receive the final confirmation, including billing details and delivery date.

3. **Admin Activities**

(a) **Log into System:** The admin logs into the system to monitor and analyze overall processes.

(b) **Monitor Number of Orders:** Admin tracks order volumes, gaining insights into demand trends and operational load.

(c) **Analyze Prophet Models:** Admin reviews the performance of demand forecasting models, such as Neural Prophet, to ensure accurate predictions and make adjustments if necessary.

(d) **View System's Overall Process:** Admin has a top-down view of system activities to ensure smooth operations and identify areas for improvement.

Here's how activity diagrams can be specifically helpful in this context:

- Enhanced Clarity of Process Flow: The diagram visually represents the process, making it easier for stakeholders to understand each role's responsibilities and interactions. By defining the steps each role follows, the diagram ensures that all

parties (Suppliers, Vendors, and Admins) understand their tasks and when each task needs to be completed.

- Identification of Key Decision Points: The activity diagram highlights critical decision points, such as the quantity check for minimum orders. This helps in identifying points where conditional logic and alternative flows come into play, ensuring that the system can handle various scenarios efficiently. For example, if the order quantity is insufficient, the system automatically checks for joint orders, thus streamlining the process without manual intervention.

- Improved Coordination Between Roles: By detailing the interactions between suppliers, vendors, and admins, the diagram promotes better coordination. Vendors can initiate joint orders when quantities are low, and suppliers can confirm orders and optimize delivery routes accordingly. This collaboration ensures that orders are fulfilled more efficiently and that each role's activities are synchronized.

- Guidance for System Design and Development: For developers, the activity diagram serves as a blueprint for creating system modules. Each activity can be mapped to specific features in the system, like order confirmation, joint order aggregation, route optimization, and demand forecasting using the Neural Prophet model. This structured approach helps in modular and organized system development, making the codebase easier to manage and maintain.

4.2 Sequence Diagram

A sequence diagram is a type of UML (Unified Modeling Language) diagram that illustrates the interactions between different components or actors in a system over time. It presents a chronological sequence of messages or actions between the various entities, providing a visual representation of the dynamic behavior of the system.

4.2.1 Key elements of Sequence Diagrams

- **Actors and Objects:** Represent the entities (users, systems, or objects) involved in the interactions.

- **Lifelines:** Vertical lines extending from actors or objects, showing their presence over time in the sequence.
- **Messages:** Arrows that represent the communication between lifelines, including method calls, data exchanges, and responses.
- **Activation Bars:** Narrow rectangles on lifelines indicating when an object is actively performing a task.
- **Control Structures:** Elements like loops, conditions, and alternative frames (alt/opt frames) that model conditional, repeated, or optional interactions.
- **Destruction:** Marked by an "X" at the end of a lifeline, indicating the termination of an object in the sequence.

4.2.2 Purpose of Sequence Diagrams

- Understanding System Behavior: Sequence diagrams help in understanding how different components or actors in a system interact with each other over time, depicting the flow of control and communication.
- Communication: They serve as a communication tool among stakeholders, including developers, designers, and project managers, by offering a clear and visual representation of system interactions.
- Design and Analysis: Sequence diagrams aid in the design and analysis phases of software development, helping to identify potential issues in the system's logic or communication flow.
- Visualizing the Workflow: It provides a clear and concise overview of the different steps involved in the Alzheimer's disease detection process.
- Identifying Potential Bottlenecks: It helps to identify any potential bottlenecks in the system, such as slow preprocessing steps or inaccurate model predictions.
- Facilitating Communication: It serves as a common language for developers, researchers, and other stakeholders involved in the project.

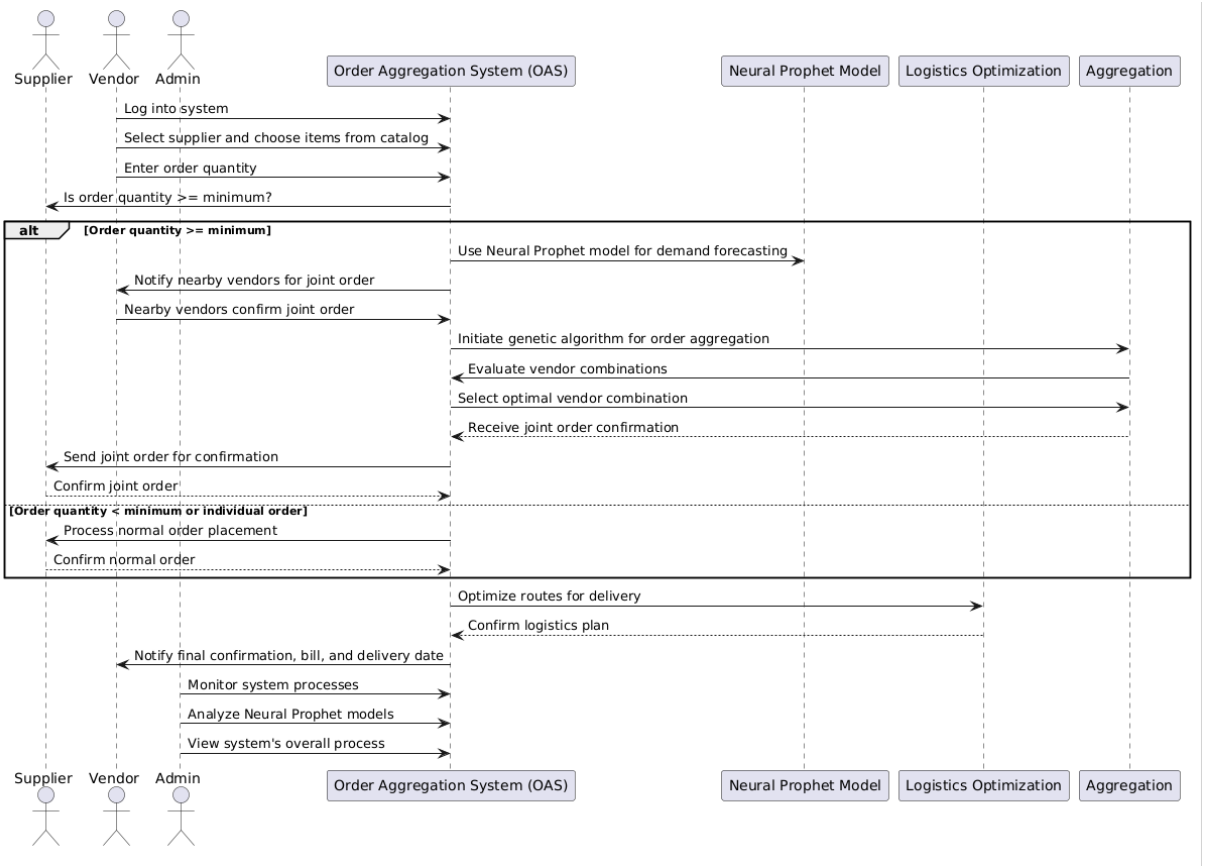

Figure 4.2: Activity Diagram

4.2.3 Key Entities and Components

- **Supplier:** Manages product inventory and fulfills confirmed orders.
- **Vendor:** Places orders for products based on the demand forecast and minimum order requirements.
- **Admin:** Monitors the overall process, analyzes model performance, and reviews logistics planning.
- **Order Aggregation System (OAS):** Core system component that manages the flow of order requests, checks conditions, and coordinates with other components.
- **Neural Prophet Model:** A demand forecasting model that helps in predicting demand and evaluating whether joint orders would be beneficial.
- **Logistics Optimization:** Component responsible for optimizing delivery routes and managing the final logistics.

4.2.4 Sequence of Interactions

1. **Login to System:** The Vendor initiates the sequence by logging into the system to access the product catalog, choose items, and place orders.

2. **Select Supplier and Choose Items from Catalog:** The Vendor selects a supplier and items from the catalog, specifying the order quantity.

3. **Order Quantity Check:** The Order Aggregation System (OAS) checks whether the order quantity meets the minimum order requirement.

    - **If Order Quantity Meets Minimum Requirement:** The order is processed as a standard, individual order. The vendor confirms the order, and it proceeds to the delivery logistics phase.
    - **If Order Quantity Is Below Minimum Requirement:** The OAS initiates steps to evaluate the feasibility of a joint order.
        (a) **Use Neural Prophet Model for Demand Forecasting:** The Neural Prophet Model is used to forecast demand and determine if combining orders from nearby vendors might help reach the minimum order quantity. This forecast provides data-driven insights into the order's potential demand over time.
        (b) **Notify Nearby Vendors for Joint Order:** If the demand forecast suggests that a joint order is viable, the OAS sends notifications to nearby vendors, inviting them to participate in a joint order.
        (c) **Nearby Vendors Confirm Joint Order:** Other vendors respond to the joint order request. If they agree, the joint order process moves forward; otherwise, the initial vendor proceeds with an individual order.
        (d) **Initiate Genetic Algorithm for Order Aggregation:** The OAS employs a Genetic Algorithm to evaluate different combinations of vendors for the joint order, aiming to find an optimal mix that maximizes efficiency and meets the minimum order requirements.
        (e) **Evaluate Vendor Combinations:** The Genetic Algorithm component iterates through various vendor combinations to find the best possible ag-

gregation. This ensures that the order is cost-effective and meets quantity requirements.

(f) **Select Optimal Vendor Combination:** The system selects the optimal vendor combination for the joint order and proceeds to confirmation.

(g) **Receive Joint Order Confirmation:** Once an optimal combination is selected, the OAS sends a joint order confirmation to all participating vendors. The joint order is now officially confirmed.

4. **Order Processing and Final Confirmation:** For both joint and individual orders, the system notifies vendors of the final confirmation, bill, and delivery date.

5. **Optimize Routes for Delivery:** The Logistics Optimization component optimizes delivery routes based on the confirmed orders, minimizing delivery time and cost by consolidating deliveries where possible.

6. **Admin Activities:** The Admin monitors system processes and performance. Admin also reviews and analyzes the accuracy of Neural Prophet Model forecasts and evaluates system functionality, using insights to make improvements to the process.

4.3 Class Diagram

A class diagram is a visual blueprint of a system in object-oriented programming, showcasing the classes, their attributes (properties), operations (methods), and the relationships between them. It's like a map that reveals the building blocks and their connections, guiding the development and understanding of the system.

4.3.1 Key Elements of Class Diagrams

- **Classes:** These are the fundamental building blocks, represented as rectangles with the class name inside. Each class encapsulates a specific concept or entity within the system, like "Customer" or "Order."

- **Attributes:** These define the data associated with a class, like "name" and "address" for the "Customer" class. They're listed within the class rectangle, often with data types specified.

- **Operations:** These represent the actions a class can perform, like "placeOrder" or "calculateTotal" for the "Order" class. They're shown as functions within the class rectangle, with their parameters and return values if applicable.
- **Relationships:** These depict the connections between classes, indicating how they interact with each other.

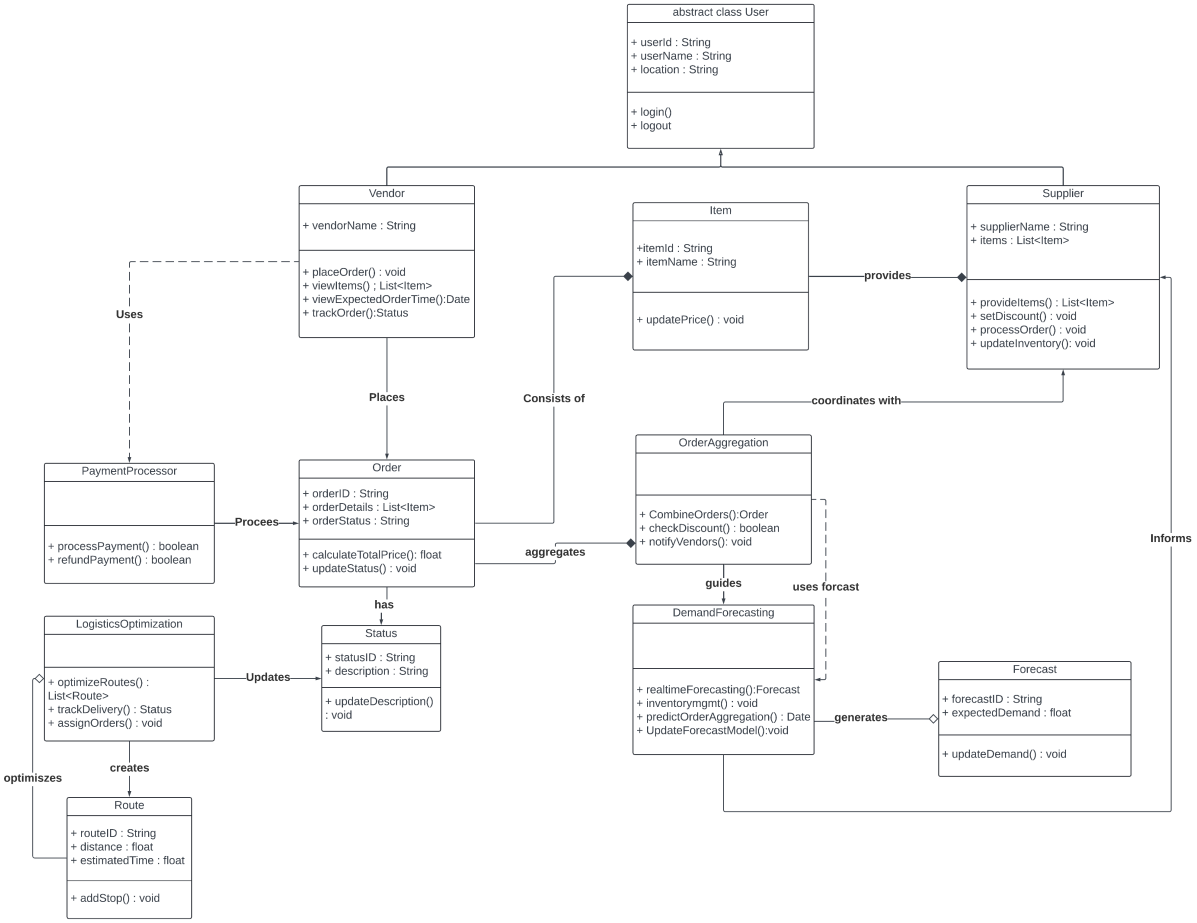

Figure 4.3: Class Diagram

4.3.2 Class Descriptions

- **User:** An abstract class representing a generic user with attributes like userId, userName, and location, along with methods login() and logout(). Inherited by Vendor and Supplier classes.
- **Vendor:** Represents a vendor who manages orders, with attributes such as vendorName and methods like placeOrder(), viewItems(), viewExpectedOrderTime(),

and trackOrder(). Associated with Order and interacts with PaymentProcessor for payment handling.

- **Supplier:** Represents a supplier managing inventory with attributes like supplierName and items, and methods like provideItems(), setDiscount(), processOrder(), and updateInventory(). Coordinates with OrderAggregation and supplies items to the Item class.

- **Item:** Represents individual items with attributes like itemId and itemName, and a method updatePrice() to modify the item's price. Associated with Order as orders consist of multiple items.

- **Order:** Represents a customer order, including attributes orderID, orderDetails (list of items), and orderStatus, with methods calculateTotalPrice() and updateStatus(). Consists of multiple Item objects and is associated with Status.

- **Status:** Represents the order's current status with attributes statusID and description, and a method updateDescription() to modify the status description. Linked to Order to provide status details.

- **OrderAggregation:** Manages combined orders with methods CombineOrders(), checkDiscount(), and notifyVendors(). Coordinates with Supplier for order processing and informs DemandForecasting about aggregate demand.

- **PaymentProcessor:** Handles payments and refunds with methods processPayment() and refundPayment(). Interacts with Vendor for transaction management.

- **LogisticsOptimization:** Optimizes delivery routes with methods optimizeRoutes(), trackDelivery(), and assignOrders(), creating Route objects as part of logistics planning.

- **Route:** Represents a delivery route with attributes routeID, distance, and estimatedTime, and a method addStop() to add stops. Created by LogisticsOptimization.

- **DemandForecasting:** Predicts demand and manages inventory with methods like realtimeForecasting(), inventoryMgmt(), predictOrderAggregation(), and update-

ForecastModel(). Uses data from Forecast to guide order aggregation and inventory planning.

- **Forecast:** Provides demand forecasts with attributes forecastID and expectedDemand, and a method updateDemand(). Generated by DemandForecasting and used for planning inventory and order aggregation.

4.3.3 Summary of Associations

1. Vendor places orders using Order and interacts with PaymentProcessor.
2. Supplier provides Items to the system and coordinates with OrderAggregation for bulk orders.
3. OrderAggregation combines orders and coordinates with Supplier, also informing DemandForecasting.
4. DemandForecasting uses forecasts from Forecast to plan inventory and order aggregation.
5. LogisticsOptimization creates Routes to optimize delivery.

4.3.4 Common Relationships

- **Association:** Shows a simple connection between two classes, like "Customer" has an "Order."
- **Aggregation:** A "has-a" relationship where one class (the whole) is made up of parts (the other class), like "Order" has "OrderItems."
- **Composition:** A stronger "has-a" relationship where the parts (the other class) cannot exist independently of the whole (one class), like "Car" has "Wheels."
- **Inheritance:** Shows a hierarchical relationship where one class (subclass) inherits properties and methods from another class (superclass), like "Employee" inherits from "Person."

4.3.5 Benefits of Using Class Diagrams

- **Clarity:** They provide a visual representation of complex systems, making them easier to understand and communicate.
- **Communication:** They serve as a common language for developers, designers, and other stakeholders to agree on the system's structure.
- **Analysis:** They help identify potential problems or inconsistencies in the system's design before implementation.
- **Documentation:** They serve as a valuable reference for understanding and maintaining the system over time.

Where You Might Encounter Them

- **Software Development:** Class diagrams are used throughout the software development lifecycle, from initial design to implementation and maintenance.
- **System Design:** They are crucial for modeling the architecture of complex systems, including web applications and enterprise software.
- **Documentation:** They are often included in technical documentation to provide a clear overview of the system's structure.

4.4 Use Case Diagram

In the Unified Modeling Language (UML), a Use Case Diagram is a sort of behavioral diagram that depicts the interactions between various actors (users or external systems) and a system that is being studied. It offers a high-level perspective on how different entities use the features or functionalities of a system. A use case diagram's main objective is to represent and illustrate the various ways users engage with a system and the results they receive.

4.4.1 Key Elements of Use Case Diagrams

- **Use Case:** A use case is an example of a particular functionality or group of related functions that the system offers to its users, also known as actors. Use cases are

designated to indicate a particular action or objective and are usually represented as ovals.

- **Actor:** An outside party interacting with the system is called an actor. Actors might be physical devices, other systems, or human actors. Stick figures are used to represent actors, who engage with the system by taking part in one or more use cases.

- **System Boundary:** Depicted as a box, the system boundary establishes the parameters of the system and demarcates its external participants. Actors reside outside the system boundaries, whereas use cases exist inside it.

- **Association:** Associations are shown as lines joining actors to use cases. These lines show that an actor participates in or communicates with a certain use case. Associations serve as a channel of communication between the use case and the actor.

- **Include Relationship:** An include relationship shows that one use case incorporates the functionality of another use case. It is represented by a dashed arrow. It suggests that the behavior of the base use case includes the added use case.

- **Extend Relationship:** An extended connection shows that, under certain circumstances, one use case can extend the behavior of another use case. It is represented by a dashed arrow with an open arrowhead. It suggests that the expanding use case is called upon under particular circumstances and is optional.

Some use cases might not include interactions with a particular actor; instead, they might represent system-wide operations or processes. We call these use cases for the system.

4.4.2 Value of Use Case Diagrams

Use Case Diagrams are valuable tools for:

- **Communication:** Use case diagrams offer a visual representation that helps stakeholders—developers, designers, and non-technical stakeholders—communicate with one other.

- **Requirements Analysis:** By recognizing and recording user-system interactions, they aid in the elicitation and clarification of system requirements.
- **System Design:** The architecture and user interfaces of the system are designed using use case diagrams as a guide.
- **Testing:** By figuring out scenarios in which users engage with the system, they can be used to generate test cases.
- **Project Planning:** By highlighting important features and their relationships, use case diagrams can help with project planning.

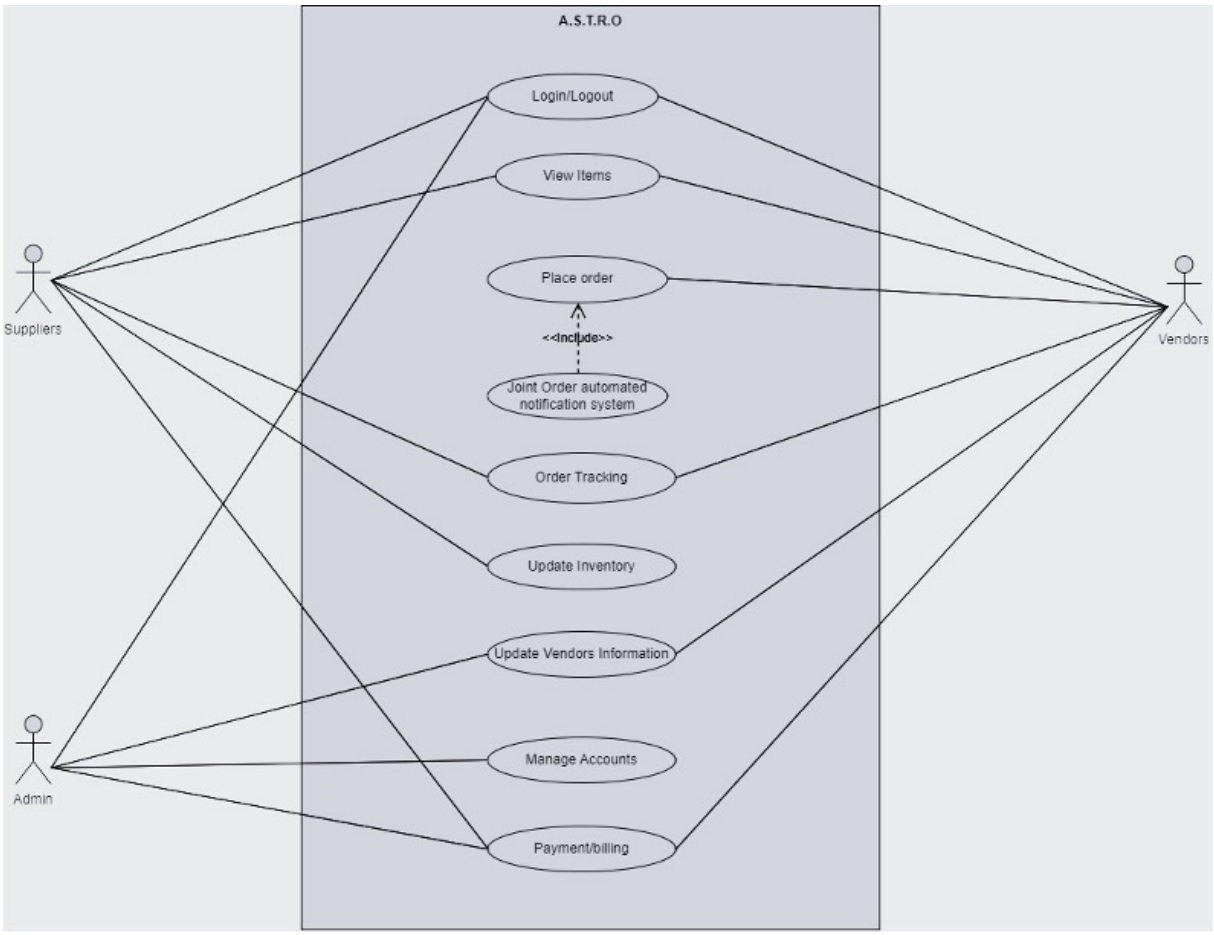

Figure 4.4: Use Case Diagram

4.4.3 Actors

1. **Suppliers:**

- Responsible for providing inventory or items for the system.
- Has access to most functions related to orders, inventory, and tracking.

2. **Vendors:**
   - Likely responsible for purchasing or receiving items in the system.
   - Similar to Suppliers, has access to most functions, especially those related to placing orders, notifications, and tracking.
3. **Admin:**
   - Responsible for overall management, including accounts and billing.
   - Has the ability to manage vendor information and accounts in addition to other functionalities.

4.4.4 Use Cases

1. **Login/Logout:**
   - All users (Suppliers, Vendors, and Admin) can log in and log out of the system, indicating that authentication is a standard feature.
2. **View Items:**
   - Suppliers and Vendors have access to this use case, allowing them to view items available in the system, possibly for ordering or tracking purposes.
3. **Place Order:**
   - Both Suppliers and Vendors can place orders in the system, which might include ordering goods or services. This use case includes an include relationship to the Joint Order Automated Notification System use case, meaning that when an order is placed, an automated notification system is triggered.
4. **Joint Order Automated Notification System:**
   - An included use case for the Place Order use case.

- This automated system sends notifications for joint orders, possibly to inform both Suppliers and Vendors of order status or details.

5. **Order Tracking:**

   - Both Suppliers and Vendors can track the status of their orders through the system, giving them visibility into order fulfillment stages.

6. **Update Inventory:**

   - This use case allows both Suppliers and Vendors to update inventory information, which could involve adding, editing, or removing items based on stock levels or requirements.

7. **Update Vendors Information:**

   - Accessible by all actors, including Admin.
   - This use case allows updating information related to Vendors, which could involve updating contact details, address, or other essential information.

8. **Manage Accounts:**

   - Specific to the Admin, who has control over managing accounts in the system.
   - This might involve user account creation, permissions, and other administrative tasks.

9. **Payment/Billing:**

   - Only the Admin has access to this functionality, likely responsible for handling the financial aspects of transactions in the system, including processing payments and issuing bills.

4.4.5 Relationships

- **Include Relationship:** The Place Order use case includes the Joint Order Automated Notification System, meaning that this automated notification is a required part of placing an order. This might help keep all parties informed about order updates.

- **System Boundary:** The shaded box labeled A.S.T.R.O represents the system boundary, meaning all the use cases within this box are functionalities provided by the A.S.T.R.O system.

Chapter 5

Results and Discussions

Chapter introduction goes here.

5.1 Section 1 Heading

Contents

5.2 Section 2 Heading

Contents

5.2.1 Subsection Heading

Contents

Chapter conclusion goes here.

Chapter 6

Conclusions & Future Scope

This section describes the conclusion of the project in one page. Write one or two paragraphs.

In this section outline the future scope/extensions possible in the project in four or five sentences.

References

[1] H. Garg and M. Dave, "Securing iot devices and securelyconnecting the dots using rest api and middleware," in *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*. IEEE, 2019, pp. 1–6.

[2] Y. Xu, J. Zhang, Q. Zhang, and D. Tao, "Vitpose++: Vision transformer for generic body pose estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

List of Publications

1. All the list of publications should be in IEEE Journal format as given in the references.

2. Publication 1

3. Publication 2

Appendix A: Presentation

Appendix B: Vision, Mission, Programme Outcomes and Course Outcomes

Vision, Mission, Programme Outcomes and Course Outcomes

Institute Vision

To evolve into a premier technological institution, moulding eminent professionals with creative minds, innovative ideas and sound practical skill, and to shape a future where technology works for the enrichment of mankind.

Institute Mission

To impart state-of-the-art knowledge to individuals in various technological disciplines and to inculcate in them a high degree of social consciousness and human values, thereby enabling them to face the challenges of life with courage and conviction.

Department Vision

Department Mission

Programme Outcomes (PO)

Engineering Graduates will be able to:

1. Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and Team work: Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.

10. Communication: Communicate effectively with the engineering community and with society at large. Be able to comprehend and write effective reports documentation. Make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

Programme Specific Outcomes (PSO)

Course Outcomes (CO)

Appendix C: CO-PO-PSO Mapping

CO - PO Mapping

CO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12
1												
2												
3												
4												
5												

CO - PSO Mapping

CO	PSO 1	PSO 2	PSO 3
1			
2			
3			
4			
5			

Justification

Mapping	Justification
CO1 - PO1	Reason
CO2 - PO2	Reason