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

This chapter describes the architecture of the system, outlining each component involved in creating an efficient demand forecasting and vendor collaboration platform. The system utilizes advanced machine learning and optimization techniques to support small-scale vendors by providing demand forecasting, order aggregation, and logistics optimization.

3.1 Dataset

The dataset is the foundation of the demand forecasting model, containing historical sales data at the store-item level. This data enables the platform to make informed predictions and optimizations.

3.2 Preprocessing

Data preprocessing is crucial to ensure that the dataset is clean, consistent, and ready for model training. This step includes handling missing values, standardizing date formats, and splitting data into training and testing sets.

3.3 Feature Extraction

Feature extraction involves deriving additional information from raw data to improve the model's predictive accuracy. The platform applies various feature engineering techniques to capture meaningful patterns in the dataset.

3.3.1 Time-Based Features

Time-based features include day-of-week, month, season, and holiday indicators. These features help capture temporal patterns and seasonality, which are important in predicting demand trends.

3.3.2 Lagged Variables

Lagged variables capture dependencies between current sales and past sales. These features are essential for understanding demand fluctuations and for integrating historical patterns into the forecasting model.

3.4 Demand Forecasting with Neural Prophet

Demand forecasting is achieved using the Neural Prophet model, which incorporates seasonality, trend components, and other covariates. This section outlines the implementation and configuration of the model to predict future sales.

3.5 Order Aggregation using Genetic Algorithm

Order aggregation is an optimization process using a Genetic Algorithm to combine multiple vendor orders. The aim is to meet minimum order quantities (MOQs) and maximize cost savings through bulk purchasing.

3.6 Logistics Optimization with Green Routing

The system's logistics optimization uses Green Routing techniques to reduce transportation costs and minimize environmental impact. This section details the two main components of the routing optimization.

3.6.1 Path Flexibility

Path flexibility allows for alternative routes within delivery schedules. It ensures efficiency in logistics by adapting routes to current traffic conditions and delivery requirements.

3.6.2 Service Time Window

Service time windows define acceptable delivery times for each location. This feature ensures timely delivery while optimizing route scheduling to meet vendor and customer needs.

3.7 Model Evaluation and Optimization

Model evaluation is performed using metrics like Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) to ensure accurate predictions. Optimization steps are undertaken to enhance model performance.

3.7.1 Evaluation

This step involves assessing the model's accuracy and performance on validation data. Evaluation metrics help identify areas for improvement in the forecasting model.

3.7.2 Hyperparameter Tuning

Hyperparameter tuning uses techniques like grid search to optimize model parameters. The goal is to enhance prediction accuracy and generalization to new data.

3.7.3 Iterative Optimization

An iterative optimization process allows the model to continuously improve by refining feature engineering and adjusting model parameters based on feedback from performance metrics.

3.8 System Integration and Notifications

The system integrates each component to ensure seamless data flow and effective communication between users. Notifications inform vendors about important updates, such as order status, inventory levels, and delivery schedules.

This chapter has outlined the system architecture, providing a detailed view of each component's role in the platform's functionality. Together, these components create a cohesive system that supports demand forecasting, order aggregation, and efficient logistics management for vendors.

—

This structure integrates your sections and subsections while describing their purpose in the system's context. Let me know if you'd like additional details in any section!

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.

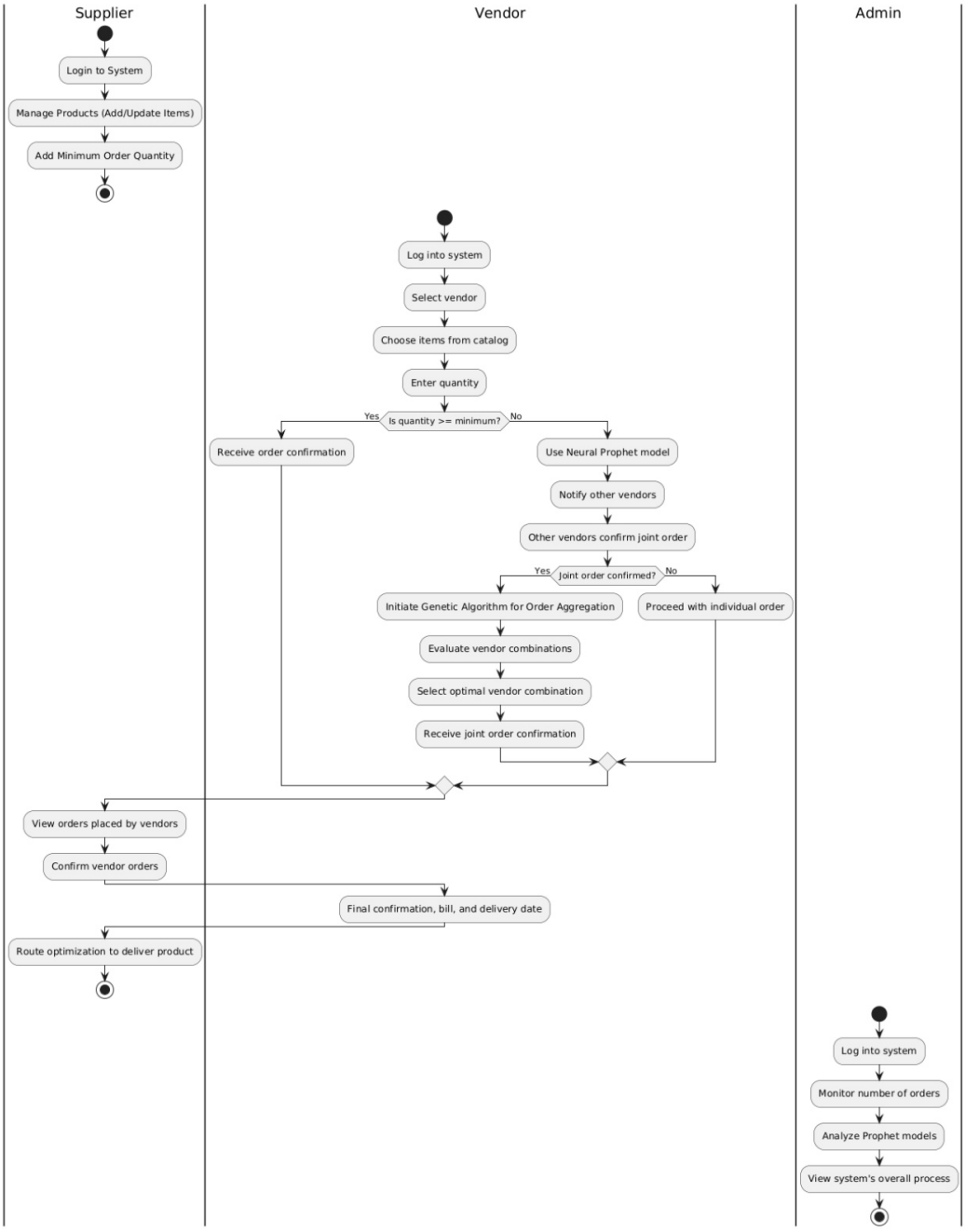

Figure 4.1: Activity Diagram

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.

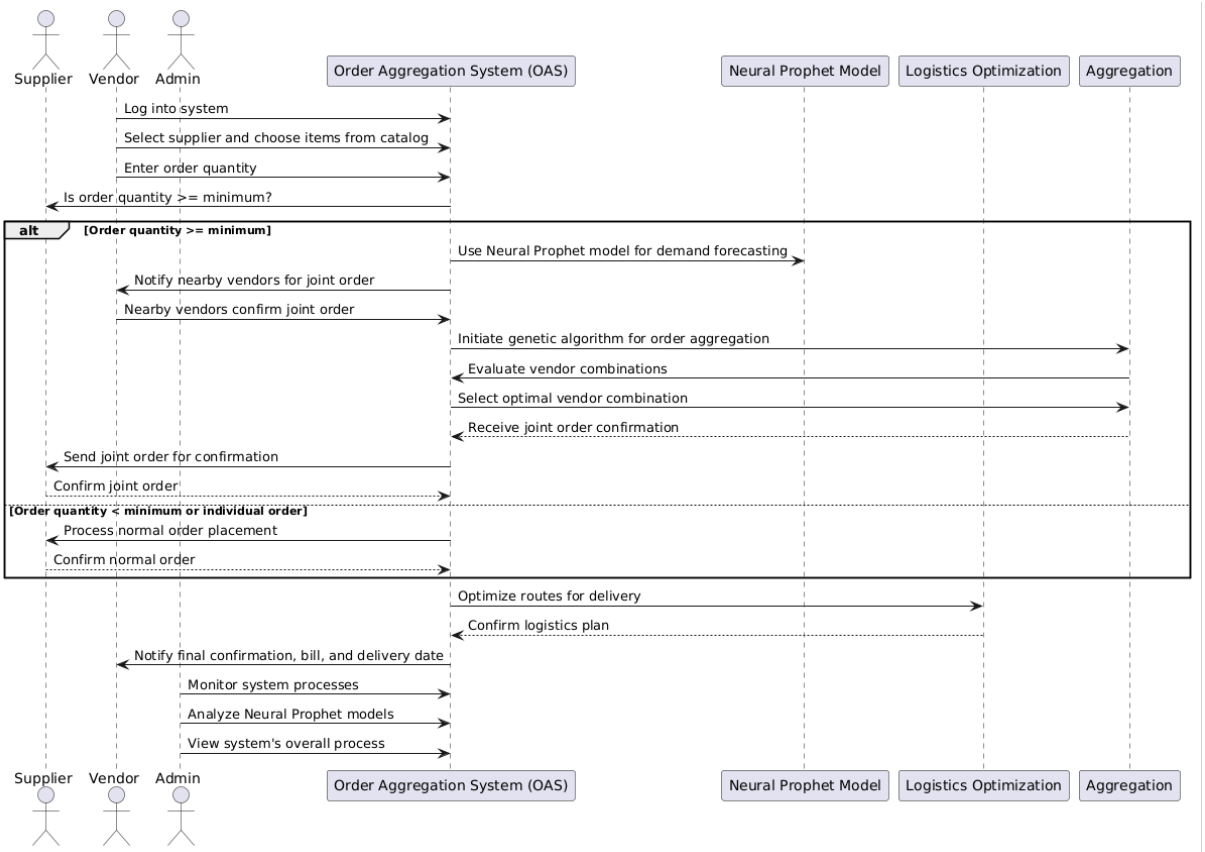

Figure 4.2: Activity Diagram

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.

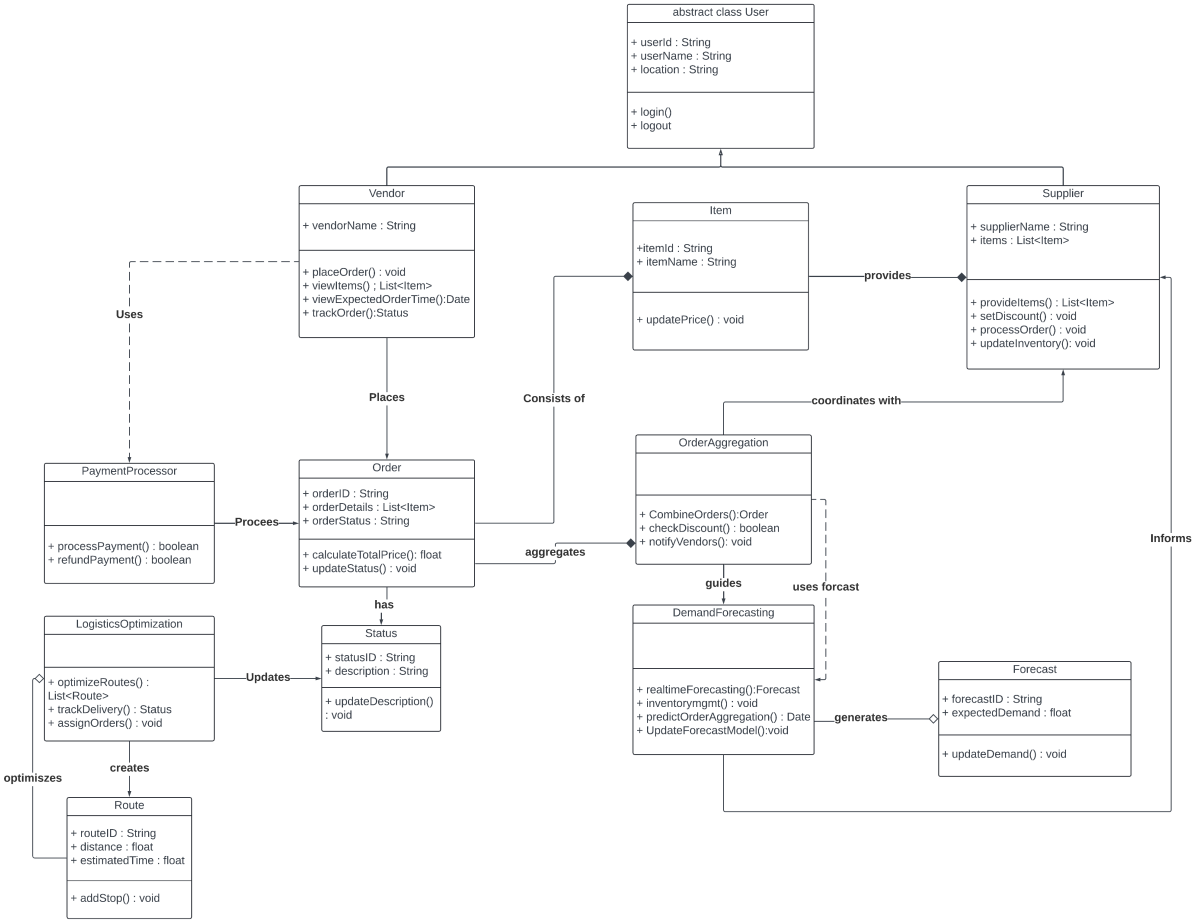

Figure 4.3: Class Diagram

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.

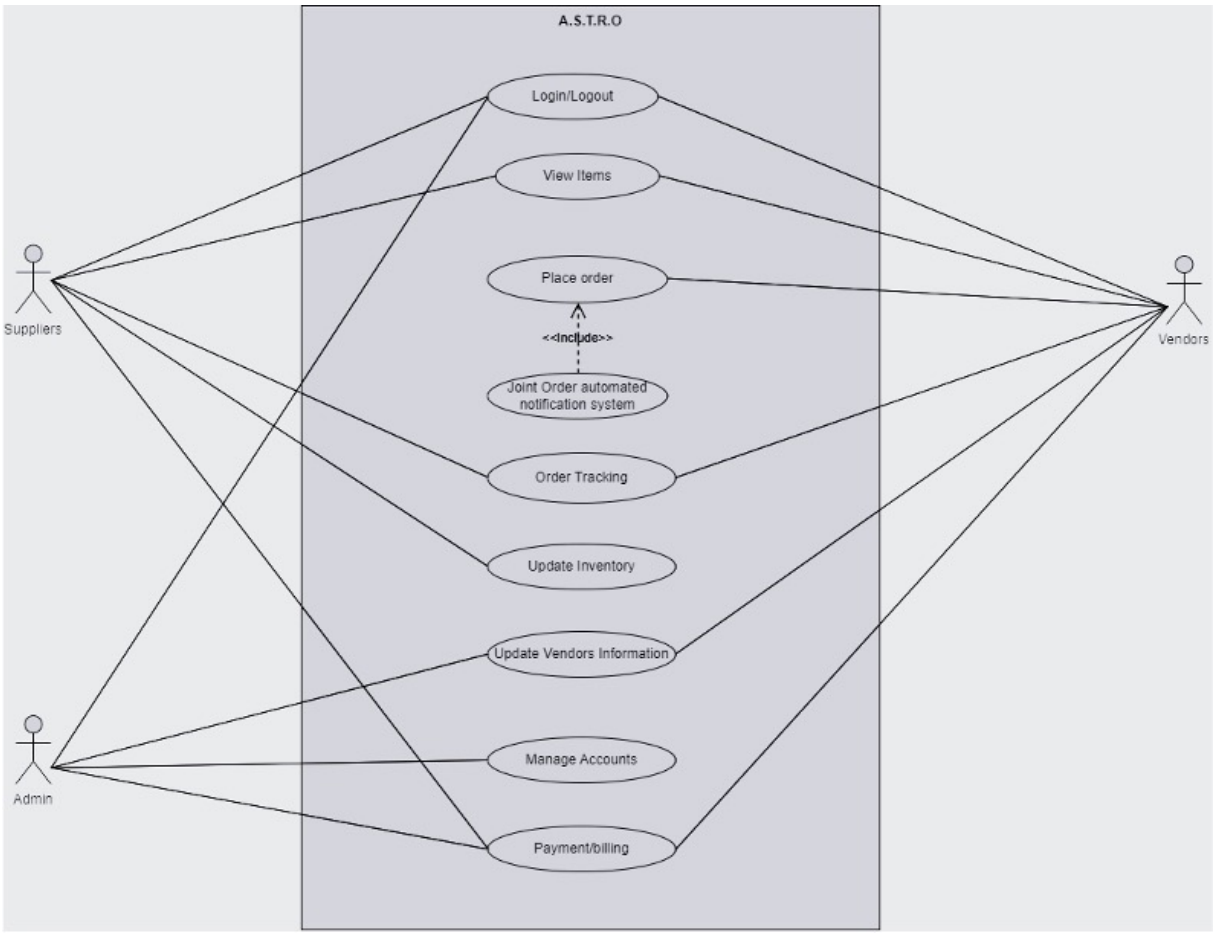

Figure 4.4: Use Case Diagram

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