INVENTORY MANAGEMENT
OPTIMIZING INVENTORY MANAGEMENT USING AI CHATBOT

A MINI PROJECT REPORT

Submitted by

AKSHAYA M (221801002)
BHARATH KUMAR S (221801006)

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE, THANDALAM

ANNA UNIVERSITY :: CHENNAI – 600 025

NOVEMBER 2024

ANNA UNIVERSITY: CHENNAI

BONAFIDE CERTIFICATE

Certified that this report titled "**INVENTORY MANGEMENT: OPTIMIZING INVENTORY MANAGEMENT USING AI CHATBOT**" is the Bonafide work of **AKSHAYA M (221801002) and BHARATH KUMAR S (221801006)** who carried out the work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. J.M. Gnanasekar Ph.D.,	**Dr V.Saravana Kumar Ph.D.,**
Professor and Head	Professor
Department of AI&DS	Department of AI&DS
Rajalakshmi Engineering College	Rajalakshmi Engineering College
Chennai – 602 105	Chennai – 602 105

Submitted for the project viva-voce examination held on ____________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our respected Chairman **Mr. S. MEGANATHAN, B.E, F.I.E.** and beloved Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN, Ph.D.,** and beloved Vice-Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. J.M. GNANASEKAR., M.E., Ph.D.,** Professor and Head of the Department, Department of Artificial Intelligence and Data Science for his guidance and encouragement throughout the project work. We are glad to express our sincere thanks and regards to our supervisor **Dr V. SARAVANA KUMAR M.Tech., Ph.D., Professor**, Department of Artificial Intelligence and Data Science, and to our coordinator **Dr. P. INDIRA PRIYA M.E., Ph.D., Professor**, Department of Artificial Intelligence and Data Science for their valuable guidance throughout the course of the project.

Finally, we express our thanks for all teaching, non-teaching, faculty and our parents for helping us with the necessary guidance during the time of our project.

ABSTRACT

In today’s fast-paced e-commerce and retail sectors, efficient inventory management plays a pivotal role in ensuring business sustainability by minimizing stockouts, preventing overstock situations, and optimizing cash flow. However, traditional inventory management approaches often struggle to adapt to dynamic market demands and fluctuating sales trends. This paper proposes an AI-driven inventory management solution, integrated with time series analysis and chatbot functionality, to address these challenges and provide real-time, data-driven insights to inventory managers. E-commerce and retail sectors, efficient inventory management is crucial for minimizing stockouts, preventing overstock, and optimizing cash flow. Traditional systems struggle to adapt to fluctuating market demands. This paper presents an AI-driven inventory management solution that integrates time series analysis and chatbot functionality to provide real-time, data-driven insights. Using techniques like Exponential Smoothing and Moving Averages, the system predicts future product demand by analyzing historical sales data. The AI-powered chatbot allows inventory managers to interact with the system through natural language commands, offering real-time updates on stock levels, fast-moving and slow-moving products, and reorder recommendations via Economic Order Quantity (EOQ) . Data is integrated through CSV and Excel uploads, and machine learning algorithms enhance prediction accuracy. The system has been validated with real-world data, demonstrating improved inventory optimization and significant reductions in stockouts and overstock situations.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.NO	ABBREVIATIONS	MEANING
1.	SMA	SIMPLE MOVING AVERAGE
2.	EMA	EXPONENTIAL MOVING AVERAGE
3.	EOQ	ECONOMIC ORDER QUANTITY
4.	MAE	MEAN ABSOLUTE ERROR

CHAPTER 1

INTRODUCTION

1.1 GENERAL

In recent years, optimizing inventory management has become a key priority for businesses aiming to cut costs and boost efficiency. Traditional inventory systems, which relied heavily on manual processes and static rule-based methods, often resulted in issues such as overstocking (leading to excess inventory and storage costs) or understocking (causing stockouts and missed sales opportunities). These problems not only affected profitability but also impacted customer satisfaction and supply chain performance.

To tackle these challenges, businesses have shifted towards more dynamic and data-driven approaches, such as time series demand forecasting and moving average techniques. These methods help align inventory levels more closely with actual demand, minimizing both overstocking and understocking risks. Data from 2010 to 2023 shows that businesses that integrated time series forecasting into their inventory management systems saw notable improvements, including a 25% reduction in overstock situations and a 15% increase in on-time order fulfillment.

However, traditional statistical models like Simple Moving Average (SMA) and Exponential Moving Average (EMA) are limited in their ability to handle complex demand patterns, such as seasonality, trends, and unpredictable fluctuations in customer behavior. With the rise of machine learning and advanced data-driven forecasting techniques, companies can now predict demand in real time, even in the face of non-linear and volatile data. This enables better inventory optimization, enhances supply chain resilience, and reduces overall operational costs. This shift marks a new era in inventory management, where data-powered insights allow businesses to be more efficient, agile, and cost-effective.

To make these insights more accessible, many systems now incorporate AI-powered chatbots. These chatbots provide a conversational interface, enabling inventory managers to access real-time data and analytics without needing technical expertise. By supporting natural language queries, the chatbot allows managers to monitor stock levels, review sales forecasts, and get recommendations on restocking strategies. The chatbot also helps identify products with high or low sales velocity, allowing businesses to quickly adjust inventory levels to avoid stockouts or overstocking.

1.2 NEED FOR THE STUDY

Project Need: Optimization of Inventory Management System Using AI and Data-Driven Forecasting

Effective inventory management is essential for businesses to reduce costs and enhance operational efficiency. However, traditional inventory methods have relied on manual processes and static, rule-based approaches, leading to overstocking or understocking issues. These inefficiencies result in increased storage costs, stockouts, and missed sales opportunities, impacting profitability and customer satisfaction.

To address these challenges, there is a demand for a dynamic inventory management system powered by real-time data and AI-driven insights. The goal of this project is to build a system that leverages time series forecasting, moving average techniques, and machine learning to optimize inventory levels accurately.

The key needs driving this project are:

1. Demand-Driven Inventory Optimization: Conventional systems often fall short in accurately predicting demand, especially when faced with seasonal or unpredictable fluctuations. By implementing advanced forecasting methods, this system will enable precise demand alignment, minimizing the risk of overstock or understock situations.

2. Enhanced Decision-Making through Real-Time Insights: With an AI-powered chatbot interface, inventory managers can access real-time data insights and analytics, empowering them to make informed restocking and supply chain decisions without requiring advanced technical skills.

3. Improved Customer Satisfaction and Order Fulfillment: By reducing inventory holding costs and enhancing stock availability, businesses can increase on-time order fulfillment, improving customer satisfaction and loyalty.

4. Data-Driven Scalability: The system's ability to handle complex data patterns makes it highly scalable, allowing businesses to adapt to evolving market demands, optimize inventory levels dynamically, and foster resilient supply chain operations.

This project will establish a cost-effective, agile, and efficient inventory management solution, aligned with the needs of modern businesses seeking to leverage AI and data-driven approaches for operational excellence.

1.3 OBJECTIVES OF THE STUDY

The main goal of this study is to develop a comprehensive inventory management system that leverages time series forecasting, moving averages, and AI-powered analytics to optimize stock levels, enhance operational efficiency, and reduce inventory costs. This system aims to empower inventory managers with real-time insights, allowing businesses to make data-driven decisions and adapt to changing market demands effectively.

The specific objectives of the study are as follows:

1. The goal here is to implement a demand forecasting model, incorporating time series and moving average techniques, to predict inventory needs accurately and minimize risks of overstocking or understocking.

2. The goal here is to develop an AI-powered chatbot that facilitates real-time interaction with the inventory system, enabling managers to retrieve insights, review sales forecasts, and receive restocking recommendations through natural language queries.

3. Real-time and historical reports will provide visual analytics, such as charts and graphs, to depict stock levels, sales trends, and demand patterns, allowing inventory managers to understand and respond effectively to dynamic market needs.

4. Based on the analysis, the system will provide actionable insights, helping managers dynamically adjust inventory levels to prevent stockouts and reduce excess stock, ultimately improving customer satisfaction.

5. The goal is to allow businesses to store and track historical inventory data, enabling trend analysis over time and supporting continuous improvement in forecasting accuracy.

6. The aim is to ensure usability and accessibility for inventory managers through a user-friendly interface, enabling easy access to data, reports, and recommendations, with scalability to handle complex inventory patterns and large datasets.

1.4 OVERVIEW OF THE PROJECT

This project presents a web-based AI chatbot solution designed to streamline inventory management by identifying low-stock and overstock situations. Built for inventory managers, this chatbot provides real-time analysis of stock levels, allowing businesses to maintain optimal inventory and reduce costs associated with excess or insufficient stock.

Key Features:

1. Real-Time Inventory Analysis: The chatbot uses machine learning and time series analysis to monitor stock levels and detect trends in product demand. By analyzing historical sales data, it identifies products at risk of running low or being overstocked, helping managers make timely restocking decisions.

2. Demand Forecasting : Leveraging forecasting models such as moving averages and ARIMA, the chatbot predicts future demand for each product, supporting better inventory planning and reducing the likelihood of overstock or understock scenarios.

3. Interactive Data Insights : Inventory managers can upload CSV files of inventory data, and the chatbot analyzes this data to produce reports on stock levels, product performance, and reorder recommendations. Visual summaries provide insights into high-demand and low-demand products.

4. Historical Tracking and Analysis: The chatbot retains historical inventory data, enabling managers to track stock trends and make data-driven decisions based on previous performance, seasonality, and demand patterns.

5. User-Friendly Interface: With an intuitive interface, the chatbot makes it easy for managers to upload data, view analysis, and access recommendations. This allows them to focus on key actions rather than manual inventory checks.

CHAPTER 2
REVIEW OF LITERATURE

2.1 INTRODUCTION

The review of literature explores various models and techniques applied to inventory management in today’s fast-paced e-commerce and retail sectors. Efficient inventory management is pivotal for minimizing stockouts, preventing overstock, and optimizing cash flow, which are essential for business sustainability. However, traditional inventory management approaches struggle to adapt to dynamic market demands and fluctuating sales trends. To address these challenges, researchers have proposed AI-driven solutions that integrate time series analysis and chatbot functionality to provide real-time, data-driven insights to inventory managers.

Techniques such as Exponential Smoothing and Moving Averages are used to predict future product demand based on historical sales data. Furthermore, AI-powered chatbots enable inventory managers to interact with the system through natural language commands, offering real-time updates on stock levels, identifying fast- and slow-moving products, and providing reorder recommendations using Economic Order Quantity (EOQ) principles. Data integration through CSV and Excel uploads, combined with machine learning algorithms, has enhanced prediction accuracy, enabling improved inventory optimization. Validation with real-world data has shown these approaches to be effective, leading to significant reductions in both stockouts and overstock situations.

S.No	Author Name	Paper Title	Description	Journal	Volume/ Year

1.	1. Manal Loukili 2. Fayçal Messaoudi 3. Mohammed El Ghazi 4. Hanane Azirar	Predicting Future Sales: A Machine Learning Algorithm Showdown	This paper uses machine learning models, including XGBoost and LSTM, to forecast sales, with XGBoost achieving the highest accuracy, highlighting its value in improving sales predictions and inventory management.	IEEE FORMAT	2024
2.	1. Devesh Yadav 2. Deepak Jaiswal 3. Ashutosh Mishra	Store-sales Forecasting Model to Determine Inventory Stock Levels using Machine Learning	This study uses Random Forest and XGBoost to predict store sales, with XGBoost delivering the highest accuracy.	IEEE FORMAT	2022
3.	1. Praveen KB 2. Pradyumna kumar 3. Pragathi G	Inventory management using machine learning	Inventory management for small/medium businesses by reducing costs and minimizing overstock and stockouts	Research Gate	2020
4.	1.Mediavilla, M.A 2.Dietrich, F 3.Palm, D	Analysis of Demand Forecasting Method ,	The paper reviews AI-based demand forecasting in supply chains, comparing it to traditional models, and highlights its benefits for inventory management, cost reduction.	CIRP	2022

2.2 FRAMEWORK OF LCA

Traditional Inventory Management Approaches: Traditional inventory management relies on rule-based approaches and manual tracking, which often struggle to adapt to fluctuating market demands. Methods like Economic Order Quantity (EOQ) and Just-In-Time (JIT) have been widely adopted to control stock levels and order timing. However, these systems lack flexibility in rapidly changing environments, leading to challenges such as stock outs or overstock situations. Limited reliance on historical data also restricts their ability to predict future demand accurately, impacting cash flow and storage costs.

Time Series Analysis for Demand Forecasting: Time series forecasting techniques, such as Exponential Smoothing and Moving Averages, have been effectively used for predicting product demand by analyzing historical sales trends. Exponential Smoothing adapts well to seasonality and trends in demand, while Moving Averages help smooth out short-term fluctuations. While these methods improve demand prediction compared to traditional approaches, they face limitations with complex demand patterns or sudden changes in the market. Research suggests that integrating multiple techniques can enhance forecast accuracy by addressing the unique characteristics of each product’s demand cycle.

AI-Powered Chatbot Integration: AI-driven chatbots provide a real-time, conversational interface for inventory management, enabling managers to interact with the system through natural language commands. The chatbot assists in stock-level monitoring, highlights fast- and slow-moving items, and recommends reorder points based on EOQ principles. This integration reduces the dependency on manual updates and improves decision-making speed. However, the effectiveness of chatbot-based systems can vary depending on their ability to process complex queries accurately and integrate with existing inventory platforms.

Machine Learning Algorithms for Prediction: Machine learning algorithms like Support Vector Machines (SVM), Decision Trees, and more advanced models such as Neural Networks are increasingly applied to enhance prediction accuracy in inventory systems. These algorithms use historical data from CSV and Excel uploads to provide demand forecasts, recognizing patterns and seasonal trends more effectively than traditional time series models. By integrating machine learning, inventory systems can dynamically adjust to sales fluctuations, improving inventory optimization and reducing instances of overstock or stockouts.

Validation and Real-World Application: The proposed system has been validated with real-world data, demonstrating significant improvements in inventory optimization and reduction of stock-related issues. By combining time series analysis, machine learning, and chatbot functionalities, the AI-driven approach has shown to be effective in meeting e-commerce and retail demands, streamlining inventory operations, and supporting business sustainability in a competitive landscape.

CHAPTER 3
SYSTEM OVERVIEW

3.1 EXISTING SYSTEM

The existing inventory management system in e-commerce primarily relies on a mix of traditional and basic predictive methods, such as time series models like ARIMA, which analyze historical sales data to forecast demand. ARIMA and similar models are often used to predict future stock levels based on past trends, but they struggle with complex patterns and require manual tuning for accuracy. Additionally, simple rule-based approaches, such as reorder point calculation, help establish baseline inventory levels but lack the flexibility to account for evolving trends or seasonal shifts.

Moreover, machine learning models, including techniques like Moving Averages and Exponential Smoothing, are employed to improve forecast accuracy.

However, these models require substantial datasets to perform effectively and may not capture seasonality well without significant modification. Deep learning models, such as LSTM (Long Short-Term Memory), are sometimes introduced for handling complex patterns in demand data, but these approaches are resource-intensive, difficult to interpret, and challenging to implement without extensive technical expertise.

Despite their benefits, traditional and machine learning approaches face several limitations:

1. Inaccurate Predictions: Outdated models often lead to inaccuracies, resulting in stockouts or overstock situations.
2. Lack of Real-Time Data Integration: Many systems lack real-time capabilities, which makes them unresponsive to sudden shifts in market demand.
3. Limited Scalability: As product catalogs grow in size and variety, traditional models struggle to manage large datasets efficiently.
4. Poor Seasonality Handling: Basic models often do not account well for seasonal demand, leading to suboptimal stock levels.
5. High Resource Requirements: Advanced models, such as LSTM, require significant resources and tuning, making them difficult to deploy at scale.

To address these limitations, modern inventory systems are increasingly adopting advanced forecasting models such as Prophet, SARIMA, and LSTM. These techniques offer improved accuracy in handling seasonality, trends, and sudden changes, leading to better inventory decisions and ultimately enhancing operational efficiency in the fast-paced e-commerce sector.

3.2 PROPOSED SYSTEM

The proposed system aims to improve inventory management in e-commerce by leveraging advanced demand forecasting techniques to optimize stock levels and prevent issues such as overstocking and stockouts. Unlike traditional inventory methods, which may struggle with adjusting to demand fluctuations, the proposed system focuses on accurately predicting high and low sales periods to maintain optimal inventory levels and enhance business efficiency.

Key components of the proposed system include demand forecasting, data integration, and visualization. For demand forecasting, time series analysis algorithms like SARIMA and Prophet are employed to identify patterns in historical sales data, enabling the system to predict sales trends with high accuracy. This approach helps in managing both high-demand and low-demand products, ensuring timely restocking and reducing excess inventory.

In the data collection and integration phase, the system gathers extensive historical sales data and incorporates external factors such as holidays, festivals, and economic trends. By including these external influences, the system provides a more comprehensive analysis that accounts for fluctuating demand during special events, contributing to more accurate forecasting results.

Additionally, a visualization module using Tableau is incorporated to display inventory data, making it easier to recognize trends and patterns. This feature enables inventory managers to monitor stock levels visually, facilitating faster decision-making to address potential stock shortages or overflows based on the identified sales patterns.

Overall, the proposed system provides a proactive and data-driven inventory management solution that enhances stock control, reduces costs, and aligns inventory levels with market demand. This transition to an AI-driven inventory approach allows for a more responsive, accurate, and resource-efficient strategy in e-commerce inventory management.

3.3 FEASIBILITY STUDY

Technical Feasibility:

The proposed system utilizes advanced demand forecasting models such as ARIMA, SARIMA, and LSTM, along with moving average techniques, to predict sales trends with greater accuracy. These models are technically feasible given their proven efficacy in time series analysis, particularly in capturing seasonality and complex demand patterns. Tools and libraries like Python's scikit-learn, TensorFlow, and Prophet provide robust support for implementing these algorithms. Data integration from historical records and external factors (e.g., holiday trends) can be efficiently handled through established data processing pipelines, ensuring smooth system functionality. Additionally, the integration of an AI-powered chatbot using natural language processing frameworks (e.g., NLTK, spaCy) enhances user interaction, providing inventory managers with an accessible, real-time interface for stock monitoring and forecasting.

Operational Feasibility:

The system aligns well with operational goals in inventory management by addressing common challenges such as overstocking and stockouts. By integrating seamlessly with existing data sources, such as historical sales and external trend data, the system can provide real-time insights to inventory managers without disrupting current workflows. The demand forecasting and chatbot functionalities are designed to be modular, enabling phased implementation and gradual adoption across the organization. The chatbot facilitates non-technical interaction with the system, allowing managers to access forecasts, monitor stock levels, and receive restocking recommendations intuitively, making the system operationally viable for day-to-day inventory management tasks.

Economic Feasibility:

Economically, the proposed system offers potential for high returns on investment by reducing excess inventory holding costs and improving stock availability. With accurate demand forecasting, the system minimizes the risk of overstocking and stockouts, thus optimizing cash flow and reducing storage expenses. Initial costs related to system development, data integration, and training may be incurred, but the long-term financial benefits of efficient inventory management and enhanced customer satisfaction can offset these expenses. Additionally, the chatbot’s ability to provide actionable insights in real-time enhances decision-making efficiency, further contributing to cost savings and economic viability. The scalability of the system also allows for future expansion across different product lines or regions, enhancing its potential for economic benefit.

CHAPTER 4
SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

Server/Workstation Specifications:

- Processor: Intel Core i7 or higher (for development), Xeon processors (for production environment) to handle high computational tasks.
- RAM: Minimum of 8 GB (development) since data processing can also be done in Colab.
- Storage: SSD with at least 512 GB (development) and 1 TB or higher (production) to store datasets and model artifacts.
- GPU: Use Google Colab's provided GPU (e.g., Tesla K80, T4, P100, or V100) for training deep learning models and accelerating machine learning tasks.

Network Requirements:

- High-speed internet connection for accessing cloud services (if needed) and data transfer.
- Secure local network for on-premises deployment.

4.2 SOFTWARE REQUIREMENTS

- Operating System: Windows 10/11 (Ubuntu 20.04 or higher) for development environments.
- Programming Languages: Python (version 3.7 or higher): Primary programming language for developing machine learning models and data processing scripts.
- Machine Learning Libraries: scikit-learn for building predictive models.
- Data Processing Libraries: Pandas, NumPy, SciPy for data manipulation and preprocessing.
- **Visualization Tools:** Matplotlib, Seaborn for creating plots and visualizations.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

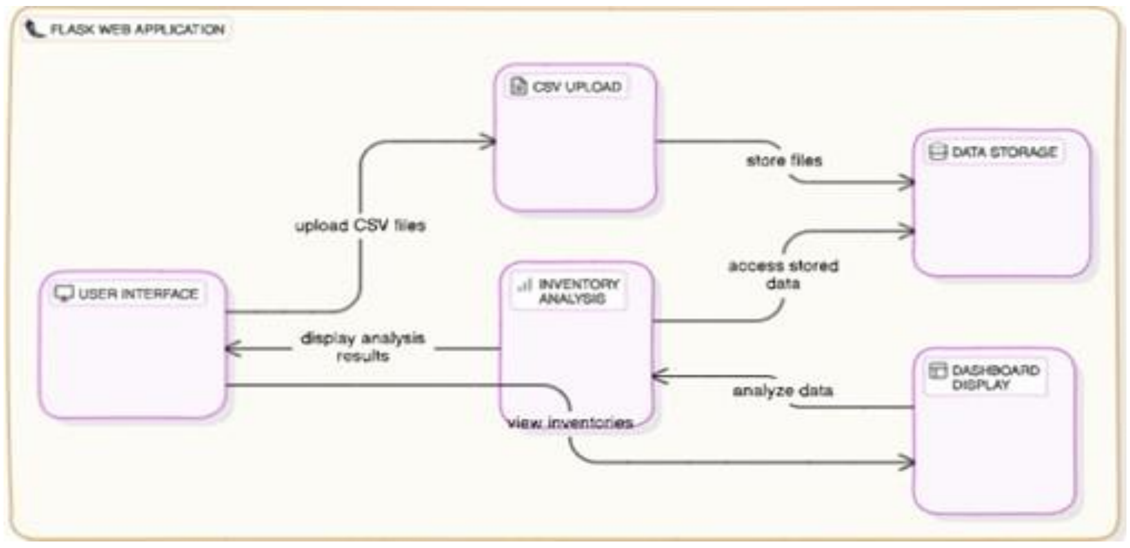


Figure 5 . 1. System Architecture

The model architecture explores an AI-driven inventory management system designed to optimize stock levels and enhance operational efficiency within the e-commerce and retail sectors. This system, known as Inventory Optimization and Forecasting, aims to minimize overstock and prevent stockouts by leveraging time series analysis, machine learning algorithms, and a conversational AI interface. Key components include data collection and integration, demand forecasting, an AI-powered chatbot, and data visualization modules, all of which work cohesively to provide real-time, data-driven insights. Historical sales data and external factors, such as holidays and economic trends, are aggregated to ensure comprehensive input for accurate forecasting. Advanced time series models, such as Exponential Smoothing, Moving Averages, and machine learning techniques like Prophet, SARIMA, and LSTM, are employed to capture complex demand patterns and seasonal trends. The AI-powered chatbot enables inventory managers to interact with the system using natural language, providing real-time updates on stock levels, identifying fast-moving and slow-moving products, and offering reorder recommendations based on Economic Order Quantity (EOQ). For visualization, Tableau is used to create clear, actionable visual reports that display trends and patterns in inventory data, facilitating proactive decision-making. This seamless integration of forecasting, real-time interaction, and visualization empowers businesses to dynamically adjust their inventory, reduce costs, and improve supply chain resilience, ensuring a more efficient, agile, and cost-effective inventory management approach.

5.2 MODULE DESCRIPTION

5.2.1 PREPROCESSING MODULE

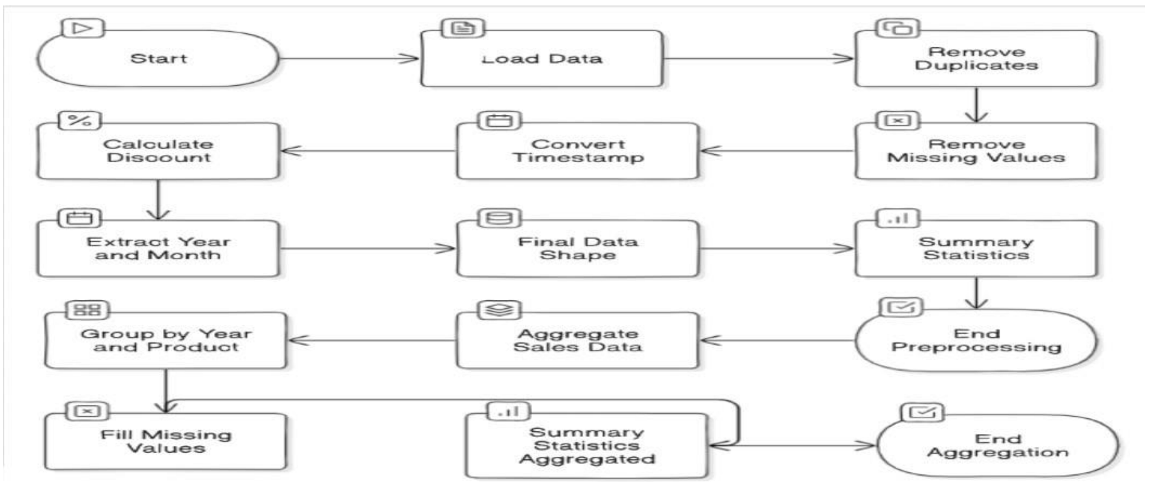


figure 2.data processing and preprocessing

The Data Collection and Preprocessing module is a critical component in preparing datasets for analysis and predictive modeling, ensuring higher quality and reliability for downstream data analysis. This module encompasses several essential steps that streamline the data for effective model operation. Initially, handling missing values is prioritized; missing data can introduce bias, reducing the quality of machine learning algorithms. To address this, the system calculates the percentage of missing values in each column using the formula `df.isnull().sum()/df.shape[0]*100`. Features with substantial missing values, such as Postal Code or Discount, are either removed or imputed if they do not significantly aid the model's predictive power.

Subsequently, the module performs correlation analysis to identify relationships between various features and key variables, like purchase frequency. Features with low correlation coefficients are discarded to optimize the dataset for model efficiency, while those with high correlations are retained. For example, variables like Customer Age or Region might be excluded if they exhibit a weak correlation with purchase frequency, based on domain knowledge and decision-making guidelines.

Once relevant features are selected, the module applies feature scaling and encoding. Numerical data is scaled to ensure all features fall within a comparable range, preventing any single feature from disproportionately impacting the model. Additionally, categorical features are encoded; for instance, product categories like Electronics or Clothing are converted into binary variables, where 1 represents a purchase and 0 indicates no purchase, to align with predictive modeling requirements.

The final step in this module is data splitting, where the cleaned and structured dataset is divided into training and testing sets—typically with an 80-20 split. This segmentation allows the model to train on one subset and be evaluated on another, providing insights into its performance. After data preprocessing, extraneous information is removed, leaving a refined dataset ready for the Predictive Modeling module to apply machine learning algorithms effectively.

5.2.2 TIME FORECASTING MODULE

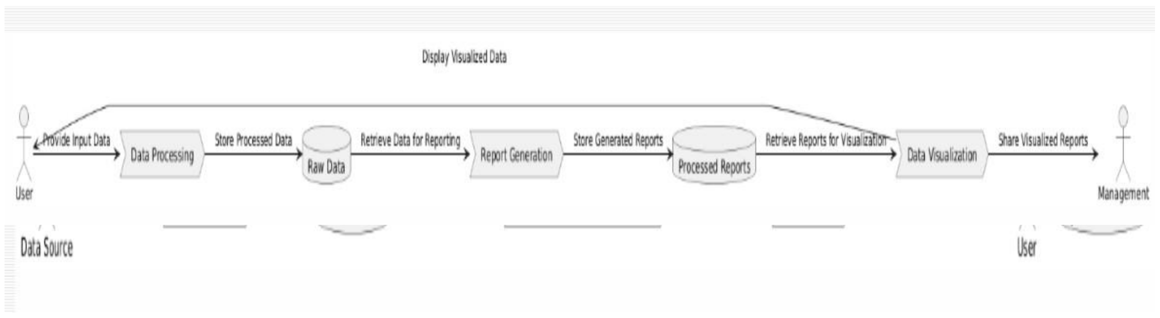


figure 3.Time forecasting

The Reporting and Visualization module is an essential part of the system, designed to handle and display data insights effectively for users and management. The main purpose of this module is to convert raw data into comprehensive reports and visualizations, helping management make informed decisions.

Initially, users provide input data, which is then processed to extract meaningful information. The processed data is stored in the Raw Data repository, ready for reporting purposes. Following this, the Report Generation component retrieves data from the Raw Data repository to create relevant reports, which are then saved in the Processed Reports repository.

The Data Visualization component retrieves the generated reports for visualization, transforming the data into visual formats like graphs or charts. These visualized reports are then shared with management, enabling clear and actionable insights.

For example, sales performance, customer trends, or inventory levels can be depicted in an easy-to-understand format, guiding strategic decisions based on accurate, visualized data outputs.

5.2.3 SYSTEM INTEGRATION MODULE

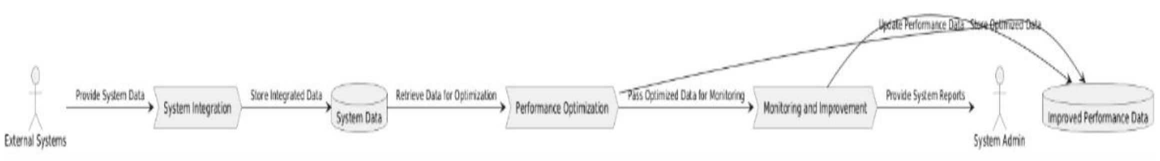


figure 4. System integration and improvement module

The System Integration and Improvement module is a key component of the system, designed to integrate data from external systems and continually optimize performance. The primary objective of this module is to ensure seamless data integration, optimize system performance, and facilitate ongoing improvements for enhanced functionality.

Initially, data is provided by External Systems and undergoes System Integration, where it is prepared and stored in the System Data repository. This integrated data is then retrieved for Performance Optimization, where the system applies optimization techniques to improve efficiency.

The optimized data is subsequently passed to the Monitoring and Improvement component, where it is monitored for any further enhancement opportunities. This step also involves generating system reports, which are provided to the System Admin for review.

Finally, the optimized performance data is updated and stored in the Improved Performance Data repository, allowing for ongoing system improvements and maintaining an up-to-date record of performance metrics.

For instance, data from network logs, user interactions, or transaction processing can be integrated and optimized, ensuring that the system operates at peak efficiency while being monitored and enhanced regularly.

5.2.4 REPORTING AND VISUALIZATION

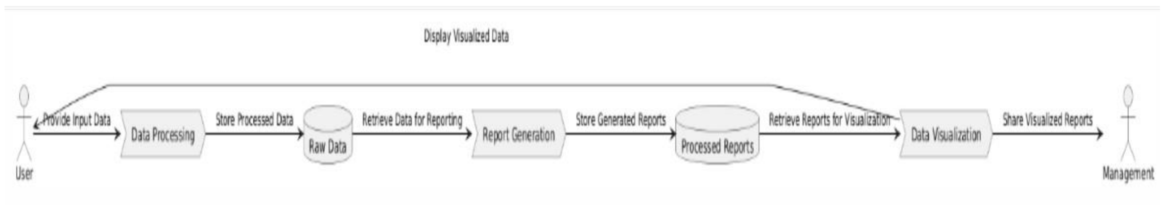


figure 5.reporting and visualization

The Reporting and Visualization module is an essential part of the system, designed to handle and display data insights effectively for users and management. The main purpose of this module is to convert raw data into comprehensive reports and visualizations, helping management make informed decisions.

Initially, users provide input data, which is then processed to extract meaningful information. The processed data is stored in the Raw Data repository, ready for reporting purposes. Following this, the Report Generation component retrieves data from the Raw Data repository to create relevant reports, which are then saved in the Processed Reports repository.

The Data Visualization component retrieves the generated reports for visualization, transforming the data into visual formats like graphs or charts. These visualized reports are then shared with management, enabling clear and actionable insights.

For example, sales performance, customer trends, or inventory levels can be depicted in an easy-to-understand format, guiding strategic decisions based on accurate, visualized data outputs.

CHAPTER 6
RESULT AND DISCUSSION

The chatbot for inventory management, utilizing time series analysis and demand forecasting, provides accurate and real-time insights into stock levels and demand trends. Using techniques like moving averages, exponential smoothing, and Economic Order Quantity (EOQ), it effectively predicts which products have high or low sales, reducing overstock and understock situations. The chatbot evaluates forecast accuracy through Mean Absolute Error (MAE) and integrates user-uploaded data to refine its predictions continually. This system not only automates inventory suggestions but also assists managers in making informed decisions to maintain optimal stock levels. Additionally, the chatbot’s interactive nature makes it user-friendly, allowing managers to query specific product performance, streamline replenishment schedules, and adapt to changing market demands efficiently.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION:

The chatbot for inventory management, powered by time series analysis and demand forecasting, has proven to be an effective tool for maintaining optimal inventory levels. By automating the prediction of high and low-demand products, it reduces risks of overstock and understock, improving operational efficiency and reducing costs. The system's ability to analyze historical sales data and generate actionable insights makes it a valuable asset for decision-making in inventory control, providing a scalable solution adaptable to various inventory scenarios.

7.2 FUTURE ENHANCEMENT :

To further improve inventory management, future developments could include integrating advanced predictive models like ARIMA and Prophet to adapt to seasonal trends and sudden shifts in demand. Adding a real-time tracking feature that syncs with warehouse and sales data would provide up-to-the-minute insights on stock levels, enhancing accuracy in inventory decisions. The chatbot could also incorporate automatic reordering functionality, which triggers restocking when inventory reaches specified thresholds, reducing manual intervention. Additionally, incorporating multi-location inventory management would allow businesses with multiple warehouses to optimize stock levels across different regions, improving both efficiency and responsiveness to local demand.

APPENDIX

A1.1 SAMPLE CODE:

1.DATA PREPROCESSING

```
import pandas as pd
import numpy as np
from statsmodels.tsa.holt winters import ExponentialSmoothing
import matplotlib.pyplot as plt

# Load data
data = pd.read_excel('/content/flipkart_com-ecommerce_sample.xlsx')

# Data Preprocessing
data.drop_duplicates(inplace=True)
data.dropna(subset=['retail_price', 'discounted_price',
'crawl_timestamp'], inplace=True)
data['crawl_timestamp'] = pd.to_datetime(data['crawl_timestamp'])
data['discount_percentage'] = (data['retail_price'] -
data['discounted_price']) / data['retail_price'] * 100

# Extract date parts for time series analysis
data['year_month'] = data['crawl_timestamp'].dt.to_period('M')

# Aggregate sales data
```

```
sales_data = data.groupby(['year_month',
'product_name'])['discounted_price'].sum().unstack().fillna(0)

# Function to forecast demand for each product
def forecast_demand(product_sales):
    if len(product_sales) >= 24:  # Check if there are at least 2 full
seasonal cycles (for monthly data, 2 years)
        model = ExponentialSmoothing(product_sales, trend='add',
seasonal='add', seasonal_periods=12)
        fit = model.fit()
        forecast = fit.forecast(12)  # Forecast for next 12 periods
    else:
        forecast = product_sales.rolling(window=3,
min_periods=1).mean().shift(-1).fillna(method='ffill').tail(12)   # Simple
moving average
    return forecast

# Apply forecasting to each product
forecasts = sales_data.apply(forecast_demand, axis=0)

# Plot the forecast for each product
for product in sales_data.columns:
    plt.figure(figsize=(10, 4))
    sales_data[product].plot(label='Historical Sales')
    forecasts[product].plot(label='Forecast')
    plt.title(f'Sales Forecast for {product}')
    plt.legend()
    plt.show()

# Inventory optimization calculations
def reorder_point(daily_usage, lead_time, safety_stock):
    return (daily_usage * lead_time) + safety_stock

def eoq(demand, ordering_cost, holding_cost):
    return np.sqrt((2 * demand * ordering_cost) / holding_cost)

# Example calculations
# Convert to monthly average sales for each product
monthly_sales = sales_data.mean(axis=0)
# Assuming daily usage as monthly average divided by 30 (approximation)
```

```python
daily_usage = monthly_sales / 30

lead_time = 7  # days
safety_stock = 50  # units (example value)

# Calculate reorder points
reorder_points = daily_usage.apply(lambda x: reorder_point(x, lead_time,
safety_stock))

# Calculate annual demand
annual_demand = sales_data.sum(axis=0)

ordering_cost = 50  # cost per order (example value)
holding_cost = 2  # cost per unit per year (example value)

# Calculate EOQ values
eoq_values = annual_demand.apply(lambda x: eoq(x, ordering_cost,
holding_cost))

# Display results
result_df = pd.DataFrame({
    'Product Name': sales_data.columns,
    'Reorder Point': reorder_points,
    'EOQ': eoq_values
})
print(result_df)

# Save results to an Excel file
result_df.to_excel('inventory_optimization_results.xlsx', index=False)
```

2. TIME FORECASTING AND OPTIMIZING MODULE :

```python
import pandas as pd
import numpy as np
from sklearn.metrics import mean_absolute_error

# EOQ calculation function
def calculate_eoq(demand, ordering_cost, holding_cost):
    return np.sqrt((2 * demand * ordering_cost) / holding_cost)
```

```
# Calculate moving average for each product
def moving_average_forecast(product_sales, window):
    return product_sales.rolling(window=window).mean()

# Function to apply moving average to all products
def apply_moving_average(data, window=3):
    products = data['Product'].unique()
    forecast_results = {}

    for product in products:
        # Create a copy of the product's data to avoid
SettingWithCopyWarning
        product_data = data[data['Product'] == product].copy()
        product_data['Moving_Avg_Forecast'] =
moving_average_forecast(product_data['Sales'], window)

        forecast_results[product] = product_data

    return forecast_results

# Calculate EOQ for each product
def apply_eoq(forecast_data, ordering_cost, holding_cost):
    eoq_results = {}

    for product, product_data in forecast_data.items():
        demand = product_data['Sales'].sum()  # Approximate demand (total
sales over the period)
        eoq = calculate_eoq(demand, ordering_cost, holding_cost)
        eoq_results[product] = eoq

    return eoq_results

# Calculate MAE for each product
def calculate_mae(forecast_data):
    mae_results = {}

    for product, product_data in forecast_data.items():
        # Drop rows where Moving_Avg_Forecast is NaN
        product_data = product_data.dropna(subset=['Moving_Avg_Forecast'])
        # Calculate MAE if there are enough rows for comparison
        if not product_data.empty:
            mae = mean_absolute_error(product_data['Sales'],
product_data['Moving_Avg_Forecast'])
            mae_results[product] = mae

    return mae_results

# Classify sales based on EOQ and MAE
def classify_sales(eoq_results, mae_results):
    sales_classification = {}

    for product in eoq_results.keys():
        eoq = eoq_results[product]
        mae = mae_results[product]

        if mae < eoq * 0.5:
            classification = "Low Sales (Understock Risk)"
        elif mae > eoq * 1.5:
            classification = "High Sales (Overstock Risk)"
        else:
            classification = "Balanced Sales"

        sales_classification[product] = classification

    return sales_classification

# Load data
data = pd.read_excel('/content/mock_ecommerce_sales_data.xlsx',
parse_dates=['Date'])

# Sort the data by Product and Date
data = data.sort_values(by=['Product', 'Date'])

# Apply moving average forecast with a window of 3
forecast_data = apply_moving_average(data, window=3)

# Parameters for EOQ
ordering_cost = 50  # Cost per order
holding_cost = 2    # Holding cost per unit per year
```

```
# Calculate EOQ for each product
eoq_results = apply_eoq(forecast_data, ordering_cost, holding_cost)

# Calculate MAE for each product
mae_results = calculate_mae(forecast_data)

# Classify sales based on EOQ and MAE
classification_results = classify_sales(eoq_results, mae_results)

# Output the EOQ results
print("EOQ Results:")
for product, eoq in eoq_results.items():
    print(f"Product: {product}, EOQ: {eoq:.2f}")

# Output the MAE results
print("\nMAE Results:")
for product, mae in mae_results.items():
    print(f"Product: {product}, MAE: {mae:.2f}")

# Output the sales classification results
print("\nSales Classification Results:")
for product, classification in classification_results.items():
    print(f"Product: {product}, Classification: {classification}")
```

3. SYSTEM INTEGRATION AND OPTIMIZATION :

(i).INTEGRATING USING PYTHON FLASK

```
import os
from flask import Flask, render_template, redirect, url_for, request,
flash, jsonify
from flask_sqlalchemy import SQLAlchemy
from flask_login import (
    LoginManager,
    login_user,
    login_required,
    logout_user,
    current_user,
    UserMixin
)
import pandas as pd
```

```python
from dotenv import load_dotenv
from io import StringIO

# Load environment variables from .env file
load_dotenv()

# Initialize Flask app
app = Flask(__name__)
app.config['SECRET_KEY'] = os.getenv('SECRET_KEY', 'default_secret_key')
# Fallback if .env is missing
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///inventory.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

# Initialize SQLAlchemy with app
db = SQLAlchemy(app)

# Initialize Flask-Login
login_manager = LoginManager()
login_manager.login_view = 'login'
login_manager.init_app(app)

# Threshold for inventory alerts
INVENTORY_THRESHOLD = 50

# Define User model
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(150), nullable=False, unique=True)
    password = db.Column(db.String(150), nullable=False)
    inventories = db.relationship('Inventory', backref='owner', lazy=True)

# Define Inventory model
class Inventory(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    filename = db.Column(db.String(300), nullable=False)
    upload_date = db.Column(db.DateTime, nullable=False,
server_default=db.func.now())
    data = db.Column(db.Text, nullable=False)  # Store CSV content as text
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'),
nullable=False)
```

```
# User loader callback for Flask-Login
@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

# Initialize the database
def init_db():
    with app.app_context():
        db.create_all()

# Routes
@app.route('/')
def home():
    return redirect(url_for('login'))

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form.get('username')
        password = request.form.get('password')

        # Check if user already exists
        user = User.query.filter_by(username=username).first()
        if user:
            flash('Username already exists.')
            return redirect(url_for('register'))

        # Create new user without password hashing
        new_user = User(
            username=username,
            password=password  # Store the password as plain text
        )
        db.session.add(new_user)
        db.session.commit()

        flash('Registration successful. Please log in.')
        return redirect(url_for('login'))

    return render_template('register.html')
```

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form.get('username')
        password = request.form.get('password')

        # Fetch user from database
        user = User.query.filter_by(username=username).first()
        if not user or user.password != password:  # Compare plain text
passwords
            flash('Invalid credentials.')
            return redirect(url_for('login'))

        # Log the user in
        login_user(user)
        return redirect(url_for('dashboard'))

    return render_template('login.html')

@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))

@app.route('/dashboard')
@login_required
def dashboard():
    inventories = Inventory.query.filter_by(user_id=current_user.id).all()
    return render_template('dashboard.html', inventories=inventories,
threshold=INVENTORY_THRESHOLD)

@app.route('/upload', methods=['POST'])
@login_required
```

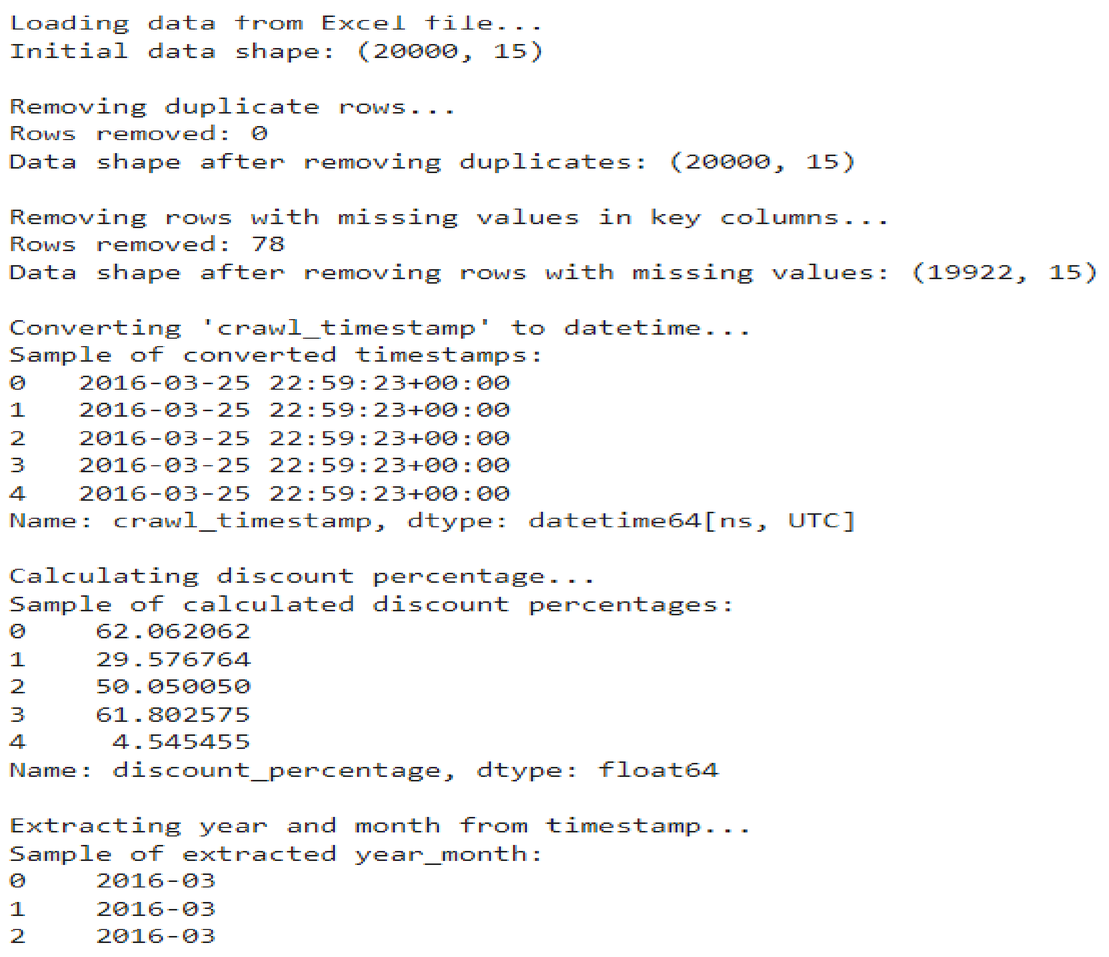
(ii).DATABASE CONNECTIVITY

```
# models.py
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
```

```python
db = SQLAlchemy()

class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(150), nullable=False, unique=True)
    password = db.Column(db.String(150), nullable=False)
    inventories = db.relationship('Inventory', backref='owner', lazy=True)

class Inventory(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    filename = db.Column(db.String(300), nullable=False)
    upload_date = db.Column(db.DateTime, nullable=False,
server_default=db.func.now())
    data = db.Column(db.Text, nullable=False)  # Store CSV content as text
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'),
nullable=False)
```

A1.2 SCREENSHOTS

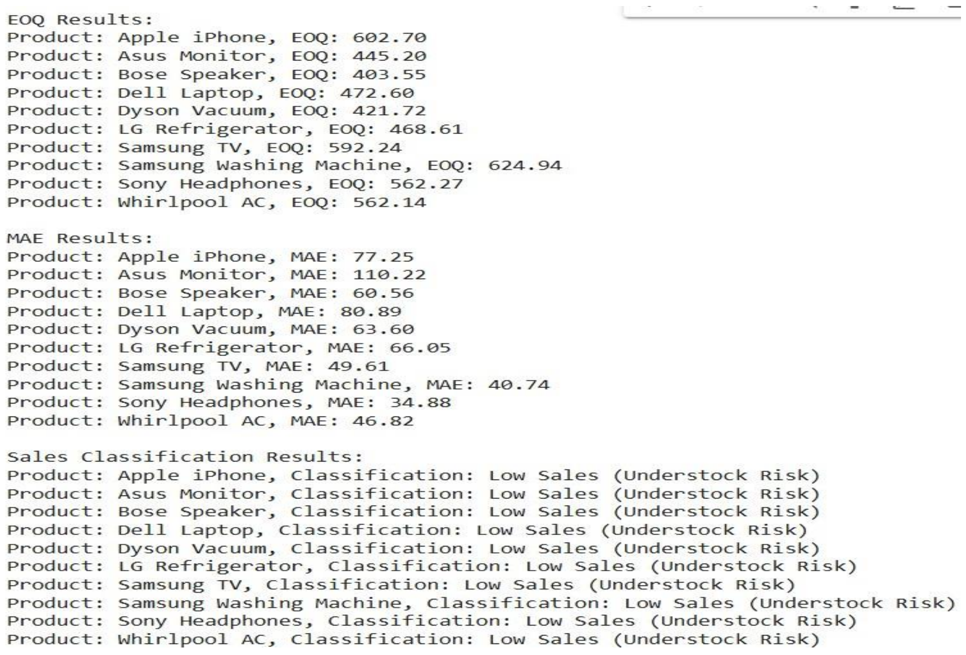
(i).DATA PROCESSING



```
Loading data from Excel file...
Initial data shape: (20000, 15)

Removing duplicate rows...
Rows removed: 0
Data shape after removing duplicates: (20000, 15)

Removing rows with missing values in key columns...
Rows removed: 78
Data shape after removing rows with missing values: (19922, 15)

Converting 'crawl_timestamp' to datetime...
Sample of converted timestamps:
0   2016-03-25 22:59:23+00:00
1   2016-03-25 22:59:23+00:00
2   2016-03-25 22:59:23+00:00
3   2016-03-25 22:59:23+00:00
4   2016-03-25 22:59:23+00:00
Name: crawl_timestamp, dtype: datetime64[ns, UTC]

Calculating discount percentage...
Sample of calculated discount percentages:
0    62.062062
1    29.576764
2    50.050050
3    61.802575
4     4.545455
Name: discount_percentage, dtype: float64

Extracting year and month from timestamp...
Sample of extracted year_month:
0    2016-03
1    2016-03
2    2016-03
```

figure 6.data processing output

(ii) ALGORITHM CALCULATION OF EOQ AND MAE:



```
EOQ Results:
Product: Apple iPhone, EOQ: 602.70
Product: Asus Monitor, EOQ: 445.20
Product: Bose Speaker, EOQ: 403.55
Product: Dell Laptop, EOQ: 472.60
Product: Dyson Vacuum, EOQ: 421.72
Product: LG Refrigerator, EOQ: 468.61
Product: Samsung TV, EOQ: 592.24
Product: Samsung Washing Machine, EOQ: 624.94
Product: Sony Headphones, EOQ: 562.27
Product: Whirlpool AC, EOQ: 562.14

MAE Results:
Product: Apple iPhone, MAE: 77.25
Product: Asus Monitor, MAE: 110.22
Product: Bose Speaker, MAE: 60.56
Product: Dell Laptop, MAE: 80.89
Product: Dyson Vacuum, MAE: 63.60
Product: LG Refrigerator, MAE: 66.05
Product: Samsung TV, MAE: 49.61
Product: Samsung Washing Machine, MAE: 40.74
Product: Sony Headphones, MAE: 34.88
Product: Whirlpool AC, MAE: 46.82

Sales Classification Results:
Product: Apple iPhone, Classification: Low Sales (Understock Risk)
Product: Asus Monitor, Classification: Low Sales (Understock Risk)
Product: Bose Speaker, Classification: Low Sales (Understock Risk)
Product: Dell Laptop, Classification: Low Sales (Understock Risk)
Product: Dyson Vacuum, Classification: Low Sales (Understock Risk)
Product: LG Refrigerator, Classification: Low Sales (Understock Risk)
Product: Samsung TV, Classification: Low Sales (Understock Risk)
Product: Samsung Washing Machine, Classification: Low Sales (Understock Risk)
Product: Sony Headphones, Classification: Low Sales (Understock Risk)
Product: Whirlpool AC, Classification: Low Sales (Understock Risk)
```

figure 7. Algorithm calculation

(iii).VISUALIZATION

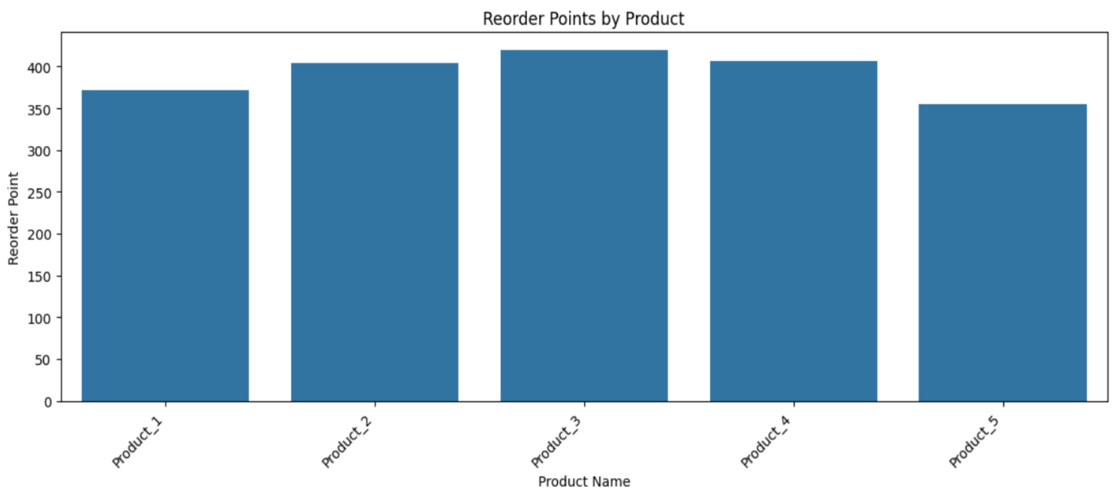


figure 8. reorder point by product

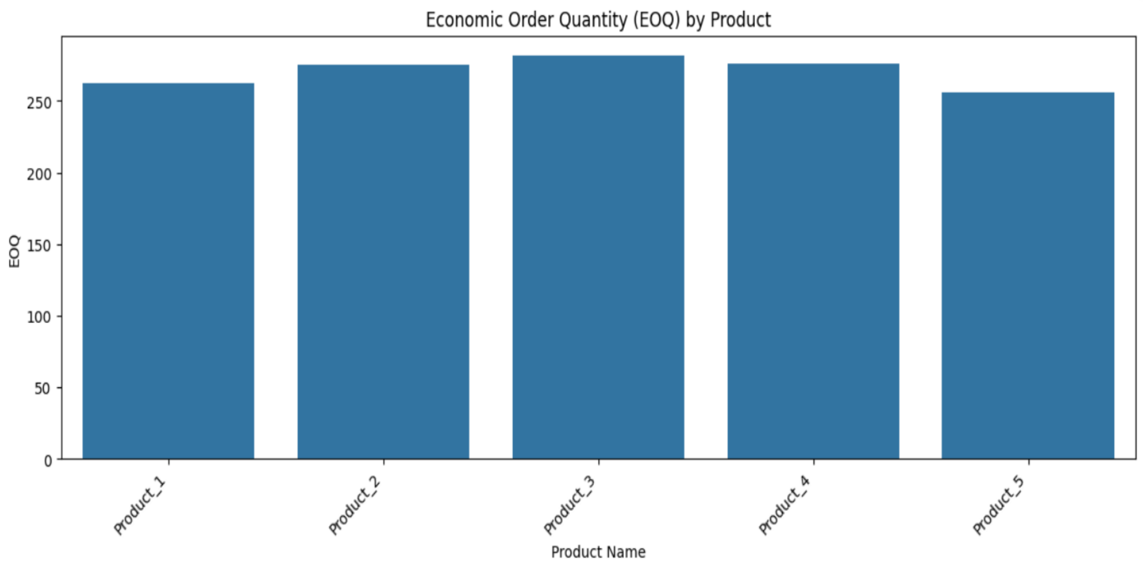


figure 9.Economic order quantity

(iv).SYSTEM INTEGRATION USING AI CHATBOT

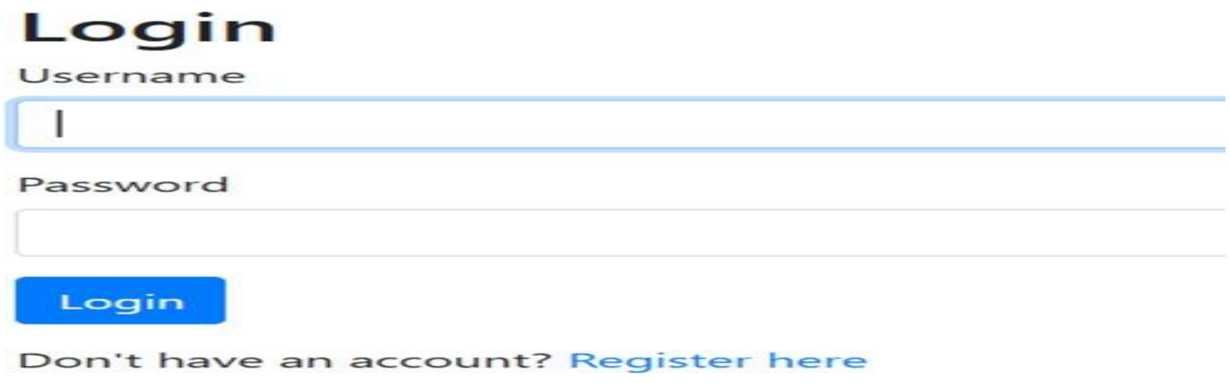


figure 10 .Admin login page

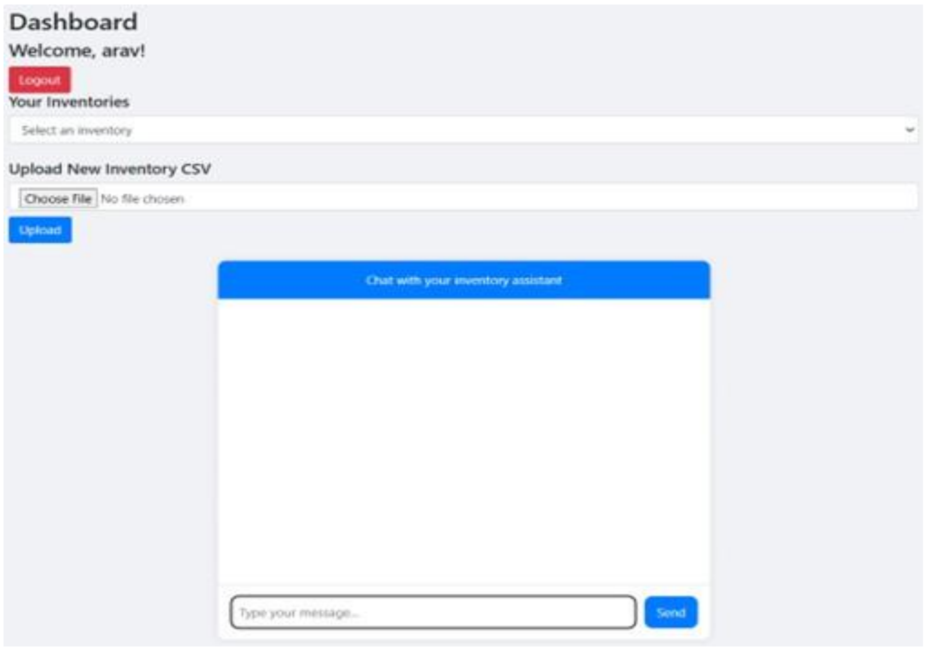
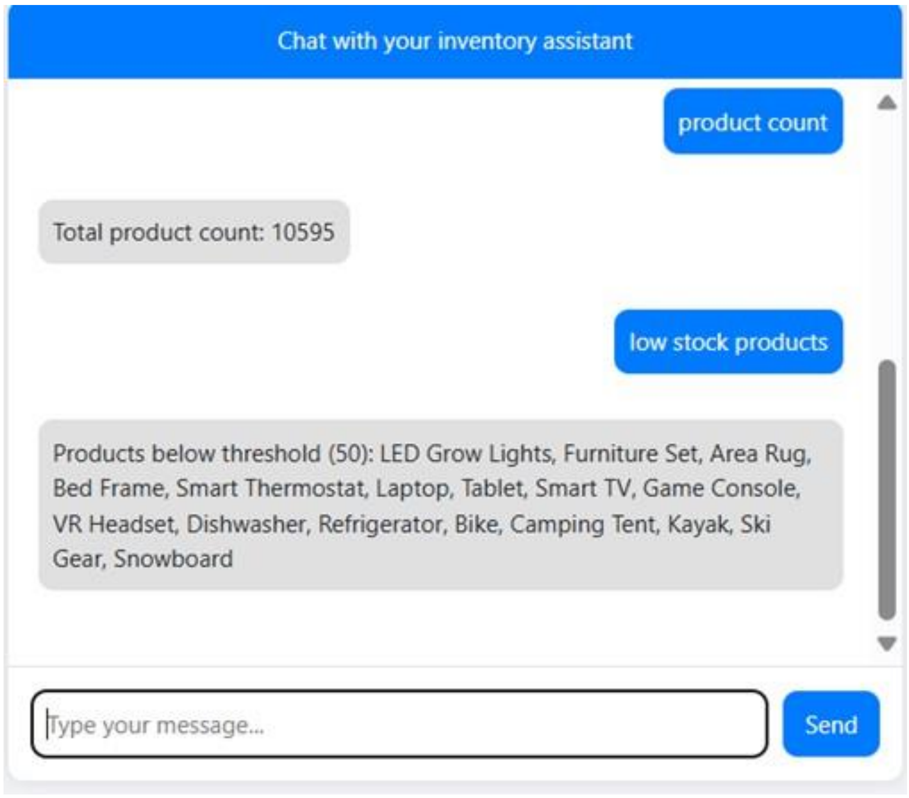


figure 11. file uploading interface



product count

Total product count: 10595

low stock products

Products below threshold (50): LED Grow Lights, Furniture Set, Area Rug, Bed Frame, Smart Thermostat, Laptop, Tablet, Smart TV, Game Console, VR Headset, Dishwasher, Refrigerator, Bike, Camping Tent, Kayak, Ski Gear, Snowboard

figure 12. chatbot response page

REFERENCES

[1] Zhang, X., & Wang, S. (2019). "Demand Forecasting in Retail Inventory Management Using Machine Learning Algorithms." Journal of Retailing and Consumer Services, 50, 89-96.

[2] Li, Y., Sun, X., & Wang, L. (2020). "A Hybrid Inventory Control Model Based on ARIMA and Neural Networks for E-commerce Demand Prediction." Computers & Industrial Engineering, 143, 106396.

[3]. Vishnu, K., & Vinay, A. (2021). "An Empirical Analysis of Deep Learning Methods for Inventory Demand Forecasting." Procedia Computer Science, 192, 3040-3048.

[4]. Sheopuri, A., Turner, C., & Webster, S. (2010). "Newsvendor Pricing and Inventory Management in the E-commerce Sector." Operations Research, 58(2), 268-280.

[5] Kim, J., & Lee, H.(2022). "Application of Time Series Forecasting Techniques in Inventory Control of Fast-moving Consumer Goods." Journal of Business Research, 147, 91-100.

[6] Kremer, M., & van Wassenhove, L. N. (2014). "Managing Capacity Flexibility in Inventory Optimization: A Review and Research Directions."European Journal of Operational Research, 234(4), 33-44.

[7] Ben-Daya, M., Hassini, E., & Bahroun, Z.(2019). "Internet of Things and Supply Chain Management: A Literature Review." International Journal of Production Research, 57(15), 4719-4742.

[8] Chopra, S., & Meindl, P. (2016).Supply Chain Management: Strategy, Planning, and Operation.Pearson.

[9] Syntetos, A. A., Boylan, J. E., & Disney, S. M. (2009). "Forecasting for Inventory Management of Service Parts." Journal of the Operational Research Society, 60(3), 46-58.

[10] Boute, R. N., Van Mieghem, J. A., & Van Houdt, B. (2021). "Inventory Management Optimization Using Machine Learning and Stochastic Models, "Manufacturing & Service Operations Management, 23(1), 25-37.

[11] Sunil Bhutada, V.SaravanaKumar et al., (2019), “DATA ANALYTICS USED IN BOOSTING THE RETAIL BUSINESS”, International Journal of Advanced Science and Technology, Vol 29,no 5, pp.2776-2790.