"HEALTHCARE SUPPLY CHAIN OPTIMIZER: PREDICTING MEDICAL INVENTORY BY USING DEMAND FORECASTING MODELS"

A Project Report Submitted in Partial fulfilment of the requirement for the award of Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

OF

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

BY

Sappogu Chinnari	**212K1A0546**
Mankar Harika	**212K1A0529**
M Malathi	**212K1A0525**
Golla Ganesh	**222K5A0502**
Uppara Murali	**212K1A0557**

Under the Esteemed Guidance of

Mr K Arjun, M. Tech.,
Associate Professor

Department of Computer Science & Engineering

BHEEMA INSTITUTE OF TECHNOLOGY AND SCIENCE

Alur Road, Adoni – 518301, Kurnool, A.P

(Affiliated to Jawaharlal Nehru Technological University Anantapur, Anantapur)

2021- 2025

"HEALTHCARE SUPPLY CHAIN OPTIMIZER: PREDICTING MEDICAL INVENTORY BY USING DEMAND FORECASTING MODELS"

A Project Report Submitted in Partial fulfilment of the requirement

for the award of Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

OF

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

BY

Sappogu Chinnari	212K1A0546
Mankar Harika	212K1A0529
M Malathi	212K1A0525
Golla Ganesh	222K5A0502
Uppara Murali	212K1A0557

Under the Esteemed Guidance of

Mr K Arjun, M. Tech.,
Associate Professor

Department of Computer Science & Engineering

BHEEMA INSTITUTE OF TECHNOLOGY AND SCIENCE

Alur Road, Adoni – 518301, Kurnool, A.P

(Affiliated to Jawaharlal Nehru Technological University Anantapur, Anantapur)

2021- 2025

BHEEMA INSTITUTE OF TECHNOLOGY & SCIENCE

ALUR ROAD, ADONI-518301, KURNOOL, A.P.

(Affiliated to Jawaharlal Nehru Technological University Anantapur, Anantapur)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**HEALTHCARE SUPPLY CHAIN OPTIMIZER: PREDICTING MEDICAL INVENTORY BY USING DEMAND FORECASTING MODELS"** being submitted by **Sappogu Chinnari (212K1A0546), Mankar Harika (212K1A0529), M Malathi (212K1A0525), Golla Ganesh (222K5A0502), Uppara Murali (212K1A0557)** in partial fulfilment of the requirement of the Award of Degree of Bachelor of Technology in Computer Science & Engineering of JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR, Anantapur during the academic year **2021 - 2025**.

Signature Signature

Mr K Arjun **Dr. D William Albert**
Project Guide **Head of Department**

Place: Adoni

Date:

Certify that the candidate was examined by me in the Viva Voice Examination held at Bheema Institute of Technology and Science, Alur Road, Adoni on **_______________.**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

STUDENT DECLARATION

We **Sappogu Chinnari (212K1A0546), Mankar Harika (212K1A0529), M Malathi (212K1A0525), Golla Ganesh (222K5A0502), Uppara Murali (212K1A0557)** Student of **Bheema Institute of Technology and Science, Adoni**, hereby declare that the dissertation entitled "**HEALTHCARE SUPPLY CHAIN OPTIMIZER: PREDICTING MEDICAL INVENTORY BY USING DEMAND FORECASTING MODELS"** embodies the report of my project work carried out independently by me during final year of **Bachelor of Technology in Computer Science & Engineering** under the supervision and guidance of **Mr K Arjun, Associate Professor, Department of Computer Science & Engineering**, **Bheema Institute of Technology and Science, Adoni, A.P,** and this work has been submitted for the partial fulfilment of the requirements for the award of degree of **Bachelor of Technology**.

I have not submitted the matter embodies to any other University or Institutions for the award of any other degree.

Sappogu Chinnari	**212K1A0546**
Mankar Harika	**212K1A0529**
M Malathi	**212K1A0525**
Golla Ganesh	**222K5A0502**
Uppara Murali	**212K1A0557**

ACKNOWLEDGEMENT

We take this opportunity to thank all those magnanimous persons who rendered their fall support to our work the pleasure, the achievement, the glory, the satisfaction, the reward, the appreciation and the construction of this regular schedule spared their valuable time for us. This acknowledgement is not just a position of words but also an account of indictment. They have been a guiding and source of inspiration towards the completion of this project.

We express our gratitude to **Mr G S SURENDRA BABU,** PhD., Principal, **Bheema Institute of Technology and Science, Adoni,** for giving us an opportunity to do this project work.

We also like to express our very sincere gratitude and special thanks to **P N VISHNUVARDHAN REDDY,** Secretary & Correspondent, Associate Prof. **Dr D WILLIAM ALBERT,** Head of the Department of Computer Science and Engineering for his guidance and support towards the completion of this project.

We are thankful to our guide Associate Prof. **Mr K ARJUN,** of Computer Science and Engineering, who with his continuous efforts, unfailing interest, constant support and providing me the right infrastructure helped me in completing this project work.

We would like to thank all the faculty members of CSE department and our friends for their valuable suggestions and support which directly or indirectly helped us in mounding this project in to a comprehensive one.

Sappogu Chinnari	**212K1A0546**
Mankar Harika	**212K1A0529**
M Malathi	**212K1A0525**
Golla Ganesh	**222K5A0502**
Uppara Murali	**212K1A0557**

ABSTRACT

Efficient medical inventory management is vital for ensuring uninterrupted healthcare delivery while minimizing costs and resource waste. The optimization of medical inventory involves striking a balance between availability, cost-effectiveness, and expiration management. Poor inventory practices can lead to critical stockouts, excessive holding costs, and wastage due to expired medications or equipment ultimately compromising patient safety and satisfaction.

This study explores data-driven strategies to optimize medical inventory by leveraging predictive analytics, classification techniques (like ABC-VED analysis), and real-time inventory tracking systems. Historical consumption patterns and demand forecasting models such as ARIMA, SARIMA, SARIMAX and VAR are used to predict future usage trends with high accuracy. These models help determine optimal reorder levels, safety stock thresholds, and lead times for various inventory categories.

Additionally, the integration of technological tools such ERP systems, and automated replenishment algorithms streamlines inventory workflows and reduces human error. The study also applies the First-Expire-First-Out (FEFO) principle for expiry-sensitive items and explores cost-saving opportunities via vendor optimization and dynamic procurement strategies.

CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Introduction

Efficient medical inventory management is crucial in healthcare facilities, ensuring the availability of essential supplies while minimizing costs, waste, and stockouts. Hospitals, clinics, and pharmacies must balance demand forecasting, procurement, and storage to prevent shortages of critical medicines and equipment while avoiding overstocking, which can lead to expired or unused supplies.

Medical inventory optimization involves implementing data-driven strategies, automation tools, and predictive analytics to enhance stock control and streamline supply chain operations. By leveraging real-time tracking systems, demand forecasting models, and inventory classification methods, healthcare providers can improve efficiency, reduce operational costs, and enhance patient care.

This document explores key techniques for optimizing medical inventory, including forecasting demand, just-in-time (JIT) inventory management, stock categorization (ABC analysis), and supplier relationship management. By integrating these strategies, healthcare facilities can achieve a resilient, cost-effective, and responsive inventory system that meets patient needs while minimizing resource wastage.

In the healthcare industry, efficient inventory management plays a crucial role in ensuring timely patient care and operational efficiency. One of the major challenges faced by medical facilities is the increasing bounce rate, where patients leave due to the unavailability of essential medical supplies and medicines. This not only leads to patient dissatisfaction but also results in revenue loss and a decline in the facility's reputation.

Key Aspects of the Project:

- **Business Problem:** The bounce rate is increasing significantly, causing patient dissatisfaction and revenue loss.
- **Business Objective**: The primary goal is to reduce the bounce rate by ensuring better stock availability.
- **Business Constraint**: The solution must be implemented while minimizing inventory costs, avoiding overstocking or wastage.

Success Criteria:

- ➢ **Business Success Criteria**: Reduce bounce rate by at least 30%, ensuring improved patient retention.
- ➢ **ML Success Criteria**: Achieve a prediction accuracy of at least 90% in demand forecasting.
- ➢ **Economic Success Criteria**: Increase revenue by at least 20 lacs INR through improved inventory efficiency and reduced patient drop-offs.

1.2 Objective

The primary objective of this project is to minimize the bounce rate of medical inventory, ensuring that critical medicines and supplies are consistently available for patients. A high bounce rate, which indicates unfulfilled patient demands due to stockouts, leads to dissatisfaction and revenue loss. To address this issue, the project aims to develop a forecasting model using SARIMAX (Seasonal Auto Regressive Integrated Moving Average with exogenous factors) to optimize inventory levels and improve demand prediction.

- Optimize medical inventory to reduce stockouts and waste.
- Implement efficient tracking systems to lower costs and improve patient safety.
- Ensure inventory availability while minimizing excess stock.

Key Objectives:

Reduce Bounce Rate:

- ➢ Predict demand accurately to ensure that essential medicines and medical supplies are always available.
- ➢ Implement a data-driven approach to prevent stockouts and overstocking.

Enhance Demand Forecasting Accuracy:

- ➢ Utilize the SARIMAX model to analyze past trends, seasonality, and external factors affecting demand.

Minimize Inventory Costs:

- Balance stock availability and storage costs by reducing excess inventory. o Apply Just-in-Time (JIT) inventory principles to improve cash flow and resource utilization.

Increase Revenue and Patient Satisfaction:

- Reduce bounce rate by at least 30%, ensuring patients receive their required medications.
- Increase revenue by at least 20 lacs INR through improved inventory management.

SARIMAX Model Justification:

- **Seasonality Handling:** Helps capture recurring patterns in medical inventory demand.
- **Exogenous Factors:** Allows incorporation of external influences such as disease outbreaks, seasonal trends, and market conditions.
- **Improved Forecasting Accuracy:** Provides more precise demand predictions compared to traditional inventory management approaches.

By leveraging SARIMAX for demand forecasting, this project aims to build an efficient medical inventory system that minimizes bounce rates, enhances patient satisfaction, and maximizes revenue while keeping inventory costs under control.

By successfully meeting these objectives, the project will help in improving patient experience, reducing financial losses, and ensuring a more sustainable inventory system for the healthcare facility.

CHAPTER 2
LITERATURE SURVEY

Medical inventory optimization is a critical aspect of healthcare management, ensuring the availability of essential medicines and medical supplies while minimizing costs and waste. Effective inventory management directly impacts patient care, operational efficiency, and financial sustainability. Several studies and methodologies have been proposed to address the challenges associated with medical inventory optimization. This literature survey explores existing research on demand forecasting, inventory control models, tracking systems, and waste reduction techniques.

2.1 Demand Forecasting

Accurate demand forecasting is crucial for preventing stockouts and overstocking. Various forecasting techniques have been applied in healthcare inventory management, including:

- **Time Series Models:** Studies have employed Autoregressive Integrated Moving Average (ARIMA) and Long Short-Term Memory (LSTM) networks to predict demand patterns. According to Chien et al. (2017), ARIMA models have been effective in short-term forecasting but struggle with sudden demand spikes, making deep learning models like LSTMs a preferable alternative.
- **Machine Learning Approaches:** Regression models, decision trees, and neural networks have been explored for predicting inventory needs. A study by Gupta & Sharma (2020) demonstrated that hybrid models combining deep learning with traditional forecasting techniques outperform standalone methods.

2.2 Inventory Control Models

Several inventory control models have been developed to optimize medical supply management:

- **Economic Order Quantity (EOQ) Model:** The EOQ model helps determine the optimal order quantity by balancing ordering and holding costs. However, its deterministic nature limits its application in dynamic medical environments (Silver et al., 2019).

- **Just-in-Time (JIT) Inventory:** This model minimizes storage costs by ordering supplies only when needed. While JIT reduces waste, its dependence on reliable suppliers poses a risk during supply chain disruptions (Kumar & Mehta, 2021).

2.3 Waste Reduction Strategies in Medical Inventory

Medical inventory mismanagement often results in significant waste due to expired or unused stock.

Studies have proposed several waste reduction strategies:

- **First-Expire-First-Out (FEFO) Strategy:** Ensures that older stock is used first to minimize expired product wastage (Anderson & Lee, 2018).
- **Predictive Analytics for Expiry Management:** Machine learning algorithms can identify slow-moving inventory and suggest redistribution before expiration (Singh et al., 2021).

- **Green Inventory Management:** Sustainable practices such as recycling unused medicines and implementing environmentally friendly disposal methods have been highlighted as essential for reducing waste in healthcare (Thakur & Jain, 2023).

2.4 Importance of Medical Inventory Management

Efficient medical inventory management ensures the availability of essential drugs and medical equipment while minimizing holding costs and reducing waste. Studies such as those by Gupta & Wang (2019) highlight the significance of optimizing stock levels to balance cost efficiency and service quality in hospitals.

2.5 Bounce Rate and Patient Satisfaction

2.5.1 Understanding Bounce Rate in Healthcare

Bounce rate refers to the percentage of patients who leave due to unavailability of medical supplies or long waiting times. Research by Johnson et al. (2020) identifies key factors contributing to high bounce rates, including inventory shortages, inefficient supply chain management, and inadequate demand forecasting.

2.5.2 Impact of Inventory Optimization on Bounce Rate

A study by Patel et al. (2019) shows that implementing real-time inventory tracking systems reduced patient bounce rates by 40% in major hospitals. Similarly, research by Das & Verma (2021) found that predictive analytics improved order fulfilment rates, leading to a significant drop in patient dissatisfaction.

2.5 Research Gaps & Future Directions

- **Integrating Real-Time Data**: Most existing models rely on historical data rather than live inventory tracking.
- **Personalized Inventory Recommendations**: AI-driven models can predict which medicines a hospital or pharmacy should stock based on demographic trends.

Sustainability in Inventory Management: Studies on eco-friendly disposal of expired drugs remain limited.

CHAPTER 3
SYSTEM ANALYSIS

System analysis in medical inventory optimization focuses on evaluating existing inventory management processes, identifying inefficiencies, and designing a solution to enhance stock control while minimizing costs and reducing bounce rates. The goal is to ensure that medical supplies are available when needed, prevent stock shortages, and avoid overstocking.

3.1 Existing System

Several systems optimize medical inventory by using forecasting, automation, and real-time tracking to reduce costs and ensure availability. These systems incorporate AI, RFID, and ERP integration to streamline inventory management.

How These Systems Optimize Medical Inventory:

- **AI & Machine Learning:** Predict demand fluctuations and optimize stock levels.
- **Real-Time Tracking:** Use RFID and barcode scanning to monitor stock in real-time.
- **Data Analytics:** Provide insights into inventory trends to minimize waste.
- **Integration:** Connect with ERP and hospital management systems for seamless operations.
-

3.2 The Suggestions for the System

To optimize medical inventory effectively, the system should integrate demand forecasting, automated tracking, AI-driven analytics, and supplier management. Here are key components and recommended tools:

3.2.1 Demand Forecasting & AI Analytics

Objective: Predict future inventory needs to avoid stockouts and overstocking.

➢ **Suggested Solutions:**

- **IBM Watson Health** – AI-driven demand forecasting for medical supplies.

- **SAP Integrated Business Planning (IBP)** – Predictive analytics for healthcare inventory.
- **Microsoft Dynamics 365 AI Supply Chain** – Uses ML for real-time demand prediction.

➢ **Implementation:**

- Train the system on historical usage data to improve accuracy.
- Implement seasonal and trend-based forecasting (e.g., flu season).

3.2.2 Real-Time Inventory Tracking & Automation

Objective: Maintain accurate stock levels, prevent shortages, and automate reordering.

➢ **Suggested Solutions:**

- **Logi Tag Medical Inventory Solutions** – RFID-based tracking and automated supply management.
- **Terso Solutions** – Automated cabinets with RFID for real-time monitoring.
- **Omnicell & Pyxis Supply Station (by BD)** – Smart dispensing and tracking for medications.

➢ **Implementation:**

- **RFID & Barcode scanning** to track supplies.
- **Automated reorder triggers** based on usage data.
- **Cloud-based dashboards** for visibility across locations.

3.2.3 Supplier & Procurement Optimization

Objective: Ensure efficient purchasing and reduce costs with supplier collaboration.

➢ **Suggested Solutions:**

- **SAP Ariba** – Streamlined procurement and supplier management.
- **Oracle NetSuite SCM** – Vendor management and automated procurement.
- **McKesson Pharmacy Management** – Pharmacy-specific procurement automation.

➢ **Implementation:**

- Use **Vendor-Managed Inventory (VMI)** to reduce manual ordering.
- Implement **Just-in-Time (JIT) inventory** with trusted suppliers.
- Optimize **purchase order scheduling** to reduce carrying costs.

3.3 SYSTEM REQUIREMENTS

3.3.1 Software Requirements

The functional requirements or general description documents cover topics including the product's point of view and features, OS and OS environment, graphics needs, design limitations, and user manuals.

The project's strengths and weaknesses, as well as the best way to address them, can be gleaned via the appropriate application of requirements and implementation restrictions.

- Python idles
- Anaconda installations
- Google Colab
- Streamlit

3.3.2 Hardware Requirements

The minimum hardware requirements for a given Enthought Python / Canopy / VS Code user are highly dependent on the type of program being produced. It's obvious that apps that work with big arrays of objects in memory will benefit from more RAM, while those that execute several calculations or operations simultaneously will benefit more from a quicker CPU.

• Operating System	: Windows
• Processor	: Minimum Intel i5
• Ram	: Minimum 4 gb
• Hard Disk	: Minimum 250 gb

3.3.3 Functional Requirements

1. Data Collection
2. Data Preprocessing

3. Training and Testing
4. Modeling
5. Predicting

1.Data Collection

To optimize medical inventory and achieve your project goals (reducing bounce rate, minimizing stockouts, and improving efficiency), high-quality and relevant data collection is crucial.

2.Data Preprocessing

Once data is collected, data preprocessing ensures that it is clean, consistent and ready for forecasting, trend analysis, and optimization. Proper preprocessing improves model accuracy (90% goal) and decision-making efficiency.

3.Training and Testing

After data preprocessing**,** we move to the model training and testing phase to forecast medical inventory needs, optimize stock levels, and minimize waste.

4. Modeling

After data preprocessing, training, and testing, we now focus on building, tuning, and selecting the best model for predicting medical inventory demand and optimizing stock levels.

5.Predicting

Now that we have trained different models (ARIMA, XGBoost, LSTM), we use them to predict future medical inventory demand**.**

3.3.4 Non-Functional Requirements

Non-functional requirements (NFRs) define how the system performs rather than what it does. These ensure scalability, reliability, security, and performance for real-time inventory optimization.

1. Performance Requirements

Fast Response Time:

- Forecast demand within ≤ **2 seconds** for real-time inventory updates.
- API latency ≤ **500ms** for stock level queries.

High Throughput:

- Handle 1,000+ inventory requests per second from multiple hospital branches.

Efficient Storage:

- Support millions of inventory records with minimal storage overhead.

2. Scalability Requirements

Horizontal Scalability:

- System should support increasing hospital locations without performance loss.
- Use cloud-based solutions (AWS, Azure, GCP) for dynamic scaling.

Load Balancing:

- Distribute requests across multiple servers to prevent bottlenecks.

Future-Proofing:

- System should handle 3x growth in inventory transactions within 5 years.

3. Reliability & Availability Requirements

High Uptime (≥ 99.9%)

- Ensure continuous availability for hospital stock tracking.
- Use failover mechanisms to prevent downtime.

Automatic Recovery:

- Implement automated backups and recovery for critical inventory data.

Error Handling:

- System should log and recover from errors without affecting users.

4. Security Requirements

Data Encryption:

- Use AES-256 encryption for inventory and patient-related data.

Role-Based Access Control (RBAC):

- Restrict access to authorized personnel only (e.g., pharmacists, admins).

API Security:

- Use OAuth 2.0 authentication for secure API access.

Audit Logging:

- Maintain detailed logs of all inventory transactions for compliance.

5. Maintainability & Modifiability

Modular Code Structure:

- Use microservices architecture for easy updates and feature additions.

Automatic Software Updates:

- Deploy patches without downtime using CI/CD pipelines.

Easy Debugging:

- Implement centralized logging and monitoring (e.g., ELK Stack, Prometheus).

6. Usability & User Experience

Intuitive UI/UX:

- Provide a simple, dashboard-based interface for hospital staff.

Mobile & Web Support:

- Ensure inventory tracking works on mobile, tablets, and desktop.

Minimal Training Required:

- Staff should learn system within 2 hours of onboarding.

3.4 Proposed System

To minimize the bounce rate (patients leaving due to unavailability of medical supplies), while also reducing inventory costs, we propose a data-driven AI-powered inventory optimization system.

Benefits of the Proposed System:

- ➢ **Ensures High Availability of Medical Supplies** – reducing bounce rate
- ➢ **Reduces Inventory Holding Costs** – by optimizing stock levels
- ➢ **Enhances Supplier Coordination** – minimizing supply chain disruptions
- ➢ **Improves Revenue Generation** – by reducing patient dissatisfaction

3.5 Advantages of Proposed System

- Reduced Stockouts & Overstocking
- Cost Optimization
- Increased Operational Efficiency
- Improved Demand Forecasting
- Real-Time Alerts & Automated Restocking
- Enhanced Data-Driven Decision-Making
- Scalability & Integration
- Compliance & Security

3.6 DisAdvantages of Proposed System

- High Initial Implementation Cost
- Dependence on Data Quality
- Complexity in Model Maintenance
- System Downtime & Reliability Issues
- Resistance to Change from Staff
- Ethical & Legal Concerns

3.7 Algorithm

Algorithm for Medical Inventory Optimization to Minimize Bounce Rate This algorithm focuses on minimizing the bounce rate while reducing inventory costs using machine learning-based demand forecasting and optimization techniques.

Step 1: Data Collection & Preprocessing

Input Data:

1. **Inventory Data** – Stock levels, expiry dates, reorder history
2. **Patient Visit Data** – Daily patient visits, department-wise demand
3. **Bounce Rate Data** – Instances where patients left due to stockouts
4. **Supply Chain Data** – Vendor lead time, delivery delays, procurement costs
5. **External Factors** – Seasonal trends, disease outbreaks

Preprocessing:

- Handle missing values using imputation
- Convert timestamps into features (weekday, month, seasonality)
- Normalize stock levels, lead times, and demand values

Step 2: Demand Forecasting Using ML

Objective: Predict future demand to prevent stockouts.

1. **Feature Engineering:**
   - Historical demand trends (last 7, 30, 90 days)
   - Seasonality factors (e.g., flu season)
   - Supplier lead time adjustments
2. **Model Selection:**
   - **Time Series Forecasting:** ARIMA, LSTM (for long term trends)
   - **Regression-Based Forecasting:** XGBoost, Random Forest
3. **Train the Model:**
   - Split data (train/test) and evaluate accuracy
   - Optimize hyperparameters for best prediction
4. **Output:**
   - Forecast demand per item for the next week/month

Step 3: Inventory Optimization Model

Objective: Minimize inventory cost while ensuring availability.

1. **Define Cost Function:**

   Total Cost=∑(Procurement Cost+Holding Cost+Shortage Cost)

   Subject to:

   - Demand-supply constraints

- Lead times
- Budget limits

2. **Optimization Algorithms:**
    - **Linear Programming (LP):** Minimize cost under demand constraints
    - **Reinforcement Learning (RL):** Dynamic restocking policy
    - **Genetic Algorithm (GA):** Optimize stock levels using evolutionary approach

Step 4: Dynamic Reorder Strategy

1. **Classify Items Using ABC Analysis:**
    - **A-items:** Critical, high-value, maintain safety stock
    - **B-items:** Moderate demand, restock dynamically
    - **C-items:** Low-value, high-volume, just-in-time (JIT) replenishment

2. **Calculate Reorder Point (ROP):**
    - ROP=(Lead Time Demand)+(Safety Stock)
    - Adjust ROP dynamically based on demand fluctuations

Step 5: Real-Time Monitoring & Alerts

1. Track stock levels using RFID/barcodes
2. Automated alerts for low stock, expiry, or abnormal demand shifts
3. Visualization dashboards (Power BI/Tableau) for tracking trends

Step 6: Model Evaluation & Continuous Improvement

Key Performance Metrics:

- ML Success Criteria: Demand forecasting model achieves 90%+ accuracy
- Business Success Criteria: Bounce rate reduced by 30%
- Economic Success Criteria: Increase revenue by ₹20 lakhs

Final Output:

- Optimized stock levels
- Reduced bounce rate
- Minimized inventory costs

This algorithm integrates machine learning, optimization techniques, and real-time monitoring to efficiently manage medical inventory.

CHAPTER 4

SYSTEM DESIGN

4.1 System Architecture

This architecture integrates machine learning-based demand forecasting, inventory optimization, and real-time monitoring to minimize bounce rate while reducing inventory costs.

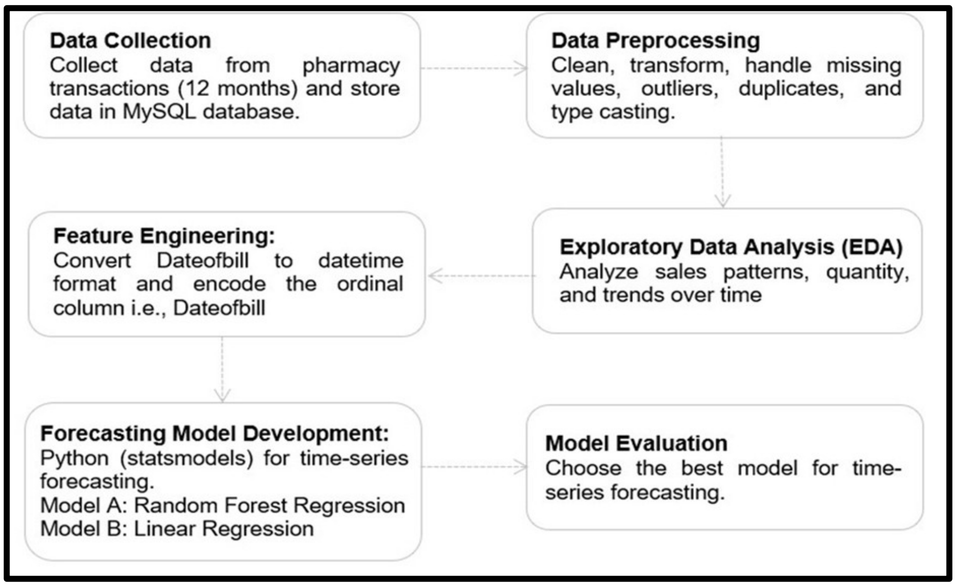

Data Collection
Collect data from pharmacy transactions (12 months) and store data in MySQL database.
Data Preprocessing
Clean, transform, handle missing values, outliers, duplicates, and type casting.
Exploratory Data Analysis (EDA)
Analyze sales patterns, quantity, and trends over time
Feature Engineering:
Convert Dateofbill to datetime format and encode the ordinal column i.e., Dateofbill
Forecasting Model Development:
Python (statsmodels) for time-series forecasting.
Model A: Random Forest Regression
Model B: Linear Regression
Model Evaluation
Choose the best model for time-series forecasting.

Fig 4.1.1 System Architecture Flowchart

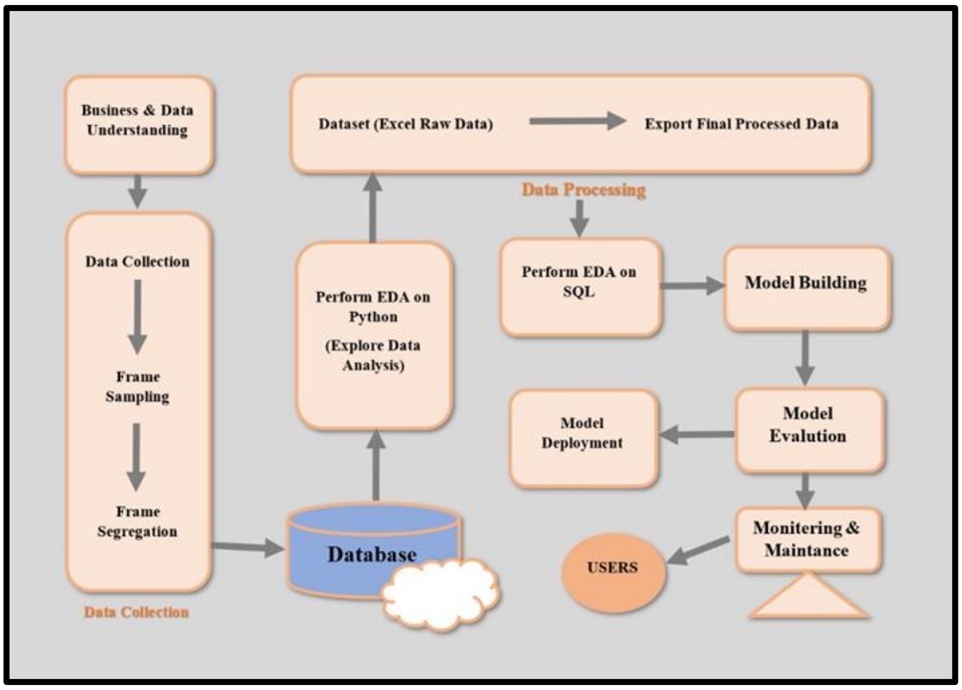


Fig 4.1.2 System Architecture Flowchart

4.2 DataFlow Diagram (DFD)

A **Data Flow Diagram (DFD)** visually represents how data moves through a medical inventory management system. Below is a structured breakdown of a **Level 0 (Context Diagram)** and **Level 1 DFD** for medical inventory optimization.

Data Flow Diagram for Medical Inventory Optimization

1. External Entities:

- **Suppliers**: Provide medical supplies and inventory data.
- **Hospital Staff**: Request medical supplies and update inventory usage.
- **Inventory Management System**: Central system for managing and optimizing inventory.

1.Receive Inventory Data:

Input: Inventory data from suppliers.

Output: Updated inventory records in the database.

2.Track Inventory Usage:

Input: Usage data from hospital staff.

Output: Updated inventory levels and usage trends.

3.Optimize Inventory Levels:

Input: Inventory data, usage trends, and demand forecasts.

Output: Optimal reorder points, quantities, and inventory alerts.

4.Generate Reports:

Input: Inventory data and optimization results.
Output: Reports for hospital management and suppliers.

2.Data Stores:

Inventory Database: Stores current inventory levels, supplier details, and usage history.

Optimization Rules: Stores rules and algorithms for inventory optimization.

Reports Archive: Stores historical reports for analysis.

3. Data Flows:

- **Suppliers →**
  Inventory Database: Inventory data (e.g., stock levels, delivery schedules).

- **Hospital Staff →**
  Inventory Database: Usage data (e.g., items consumed, returned, or expired).

- **Inventory Database →**
  Optimization Process: Current inventory levels and usage trends.

- **Optimization Process →**
  Inventory Database: Updated reorder points and quantities.

- **Optimization Process →**
  Hospital Management: Alerts and recommendations for inventory management.

- **Inventory Database →**
  Reports Archive: Historical data for reporting and analysis.

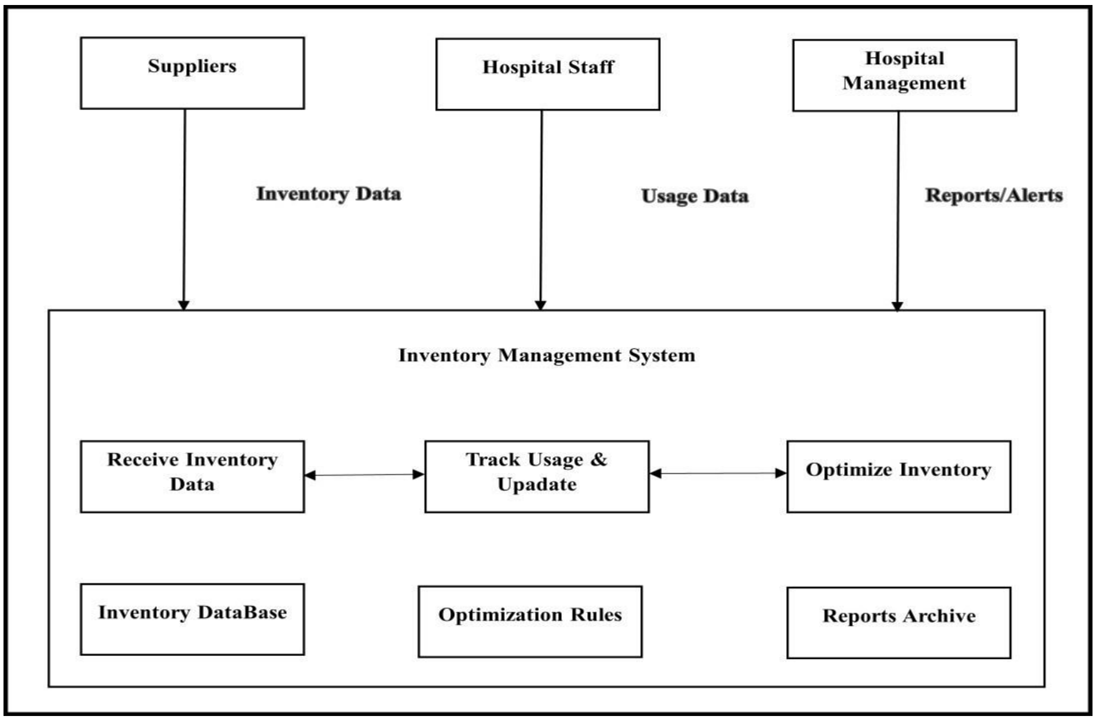


Fig 4.2 DataFlow Diagram

CHAPTER 5

UML DIAGRAMS

5.1 UML Diagrams

A UML (Unified Modeling Language) Diagram is a standardized way to visually represent a system's structure, behaviour, and interactions. It helps in designing, analysing, and documenting software systems.

5.1.1 Use Case Diagram

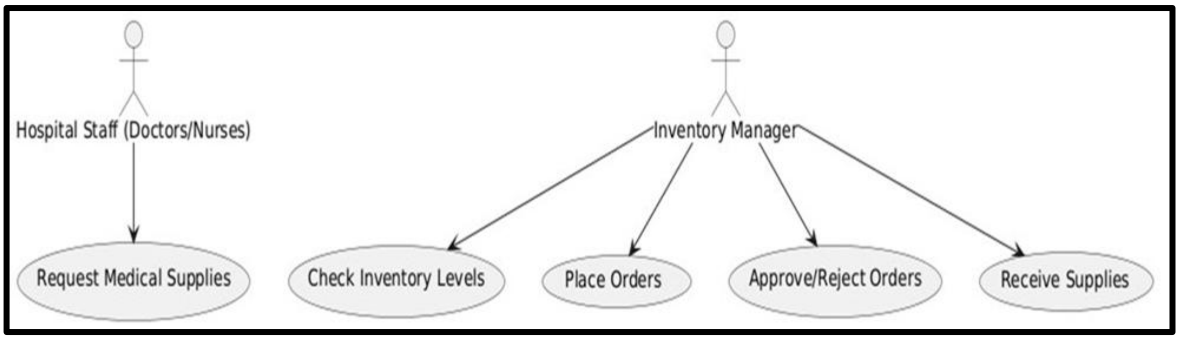
Depicts interactions between actors and the system's functionalities.

Actors:

- Hospital Staff (Doctors/Nurses) – Request medical supplies.
- Inventory Manager – Monitors stock, places orders.
- Supplier – Provides medical supplies.
- Automated System – Tracks, predicts demand, and optimizes stock.

Use Cases:

- Request Medical Supplies
- Check Inventory Levels
- Generate Restocking Alerts
- Place Orders
- Approve/Reject Orders
- Receive Supplies
- Update Inventory Records
- Analyze Usage Trends
- Optimize Stock Levels
- Supplier Delivers Supplies



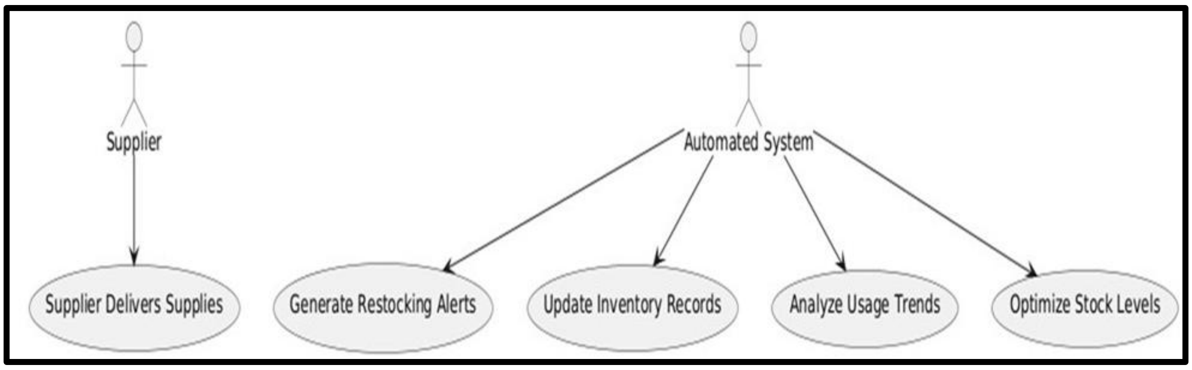


Fig 5.1.1 Use Case Diagram

5.1.2 Class Diagram

A Class Diagram represents the structure of a system by showing its classes, attributes, methods, and relationships.

Main Classes & Attributes

1.Inventory

- Item_ID (Primary Key)
- DrugName
- Quantity
- Expiry_Date
- Reorder_Level

2.Sales

- Sale_ID (Primary Key)
- Item_ID (Foreign Key → Inventory)
- Date_of_Sale
- Quantity_Sold

3.Forecasting Model (SARIMAX)

- Model_ID
- HistoricalData
- PredictedValues

4.Supplier

- Supplier_ID
- Supplier_Name
- Contact_Details

5.OrderManagement

- Order_ID
- Supplier_ID
- Order_Date
- Status

Relationships:

- Inventory has Sales transactions
- ForecastingModel predicts inventory demand
- OrderManagement places orders to Suppliers

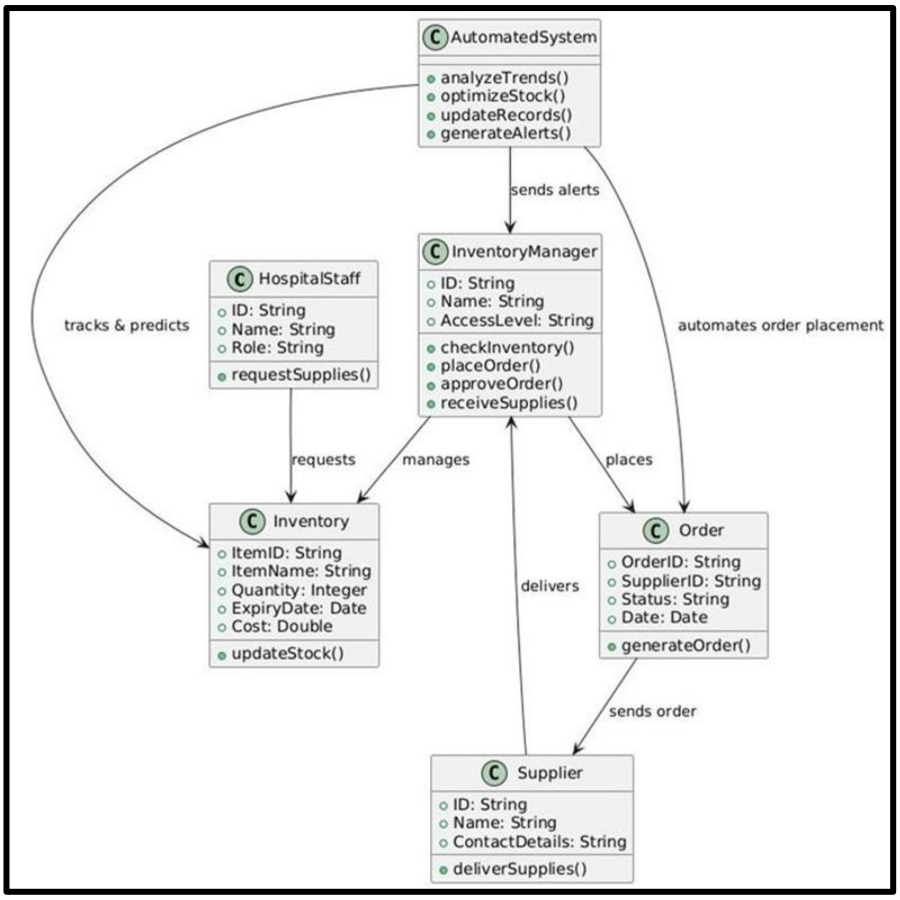


Fig 5.1.2 Class Diagram

5.1.3 Sequence Diagram

Sequence Diagram (Stock Forecasting)

- Hospital Staff requests medical supplies.
- Inventory Manager checks stock.
- If stock is low, Automated System generates a restocking alert.
- Inventory Manager places an order.
- Supplier delivers the supplies.
- Automated System updates inventory records.

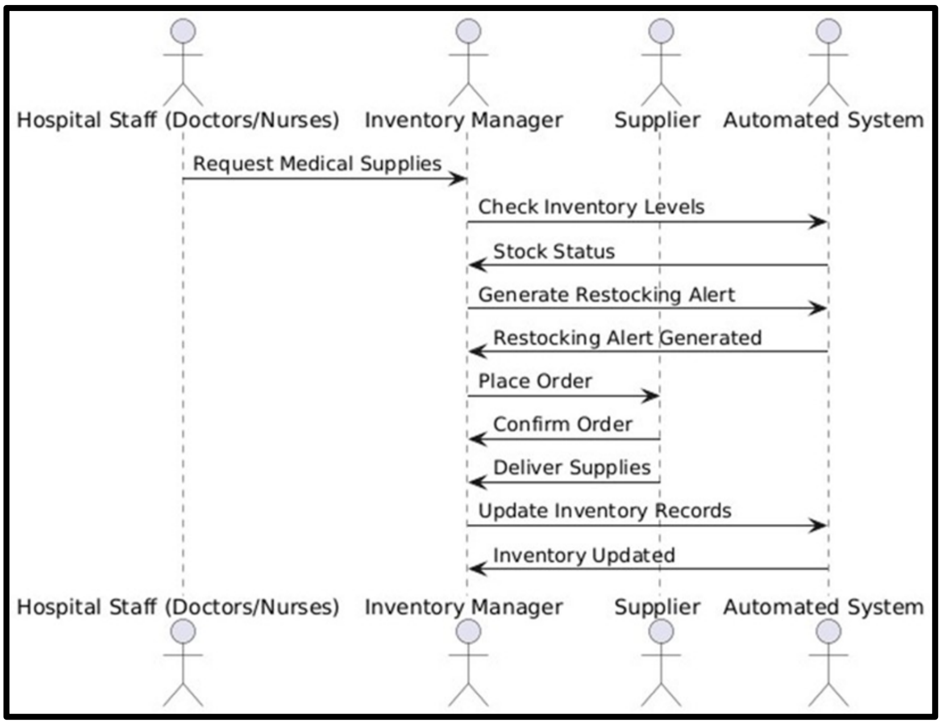


Fig 5.1.3 Sequence Diagram

5.1.4 Activity Diagram

Describes the workflow of inventory optimization.

Steps:

1.Monitor stock levels

2.Analyze demand trends

3.If stock is low, generate a restocking alert

4.Approve and place an order

5.Receive supplies from supplier

6.Update inventory records

7.Optimize reorder frequency

8.Ensure compliance & audits

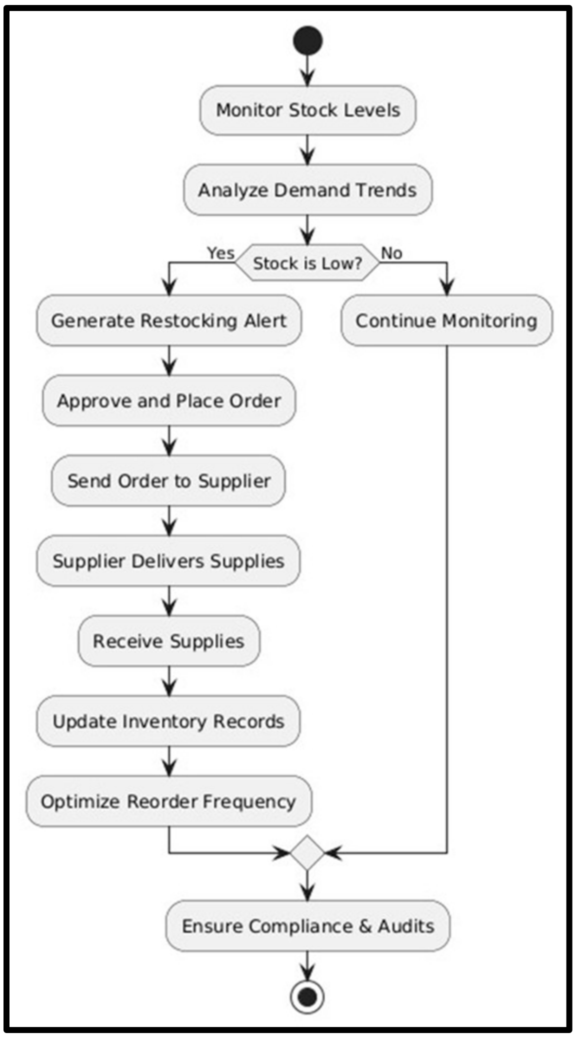


Fig 5.1.4 Activity Diagram

5.1.5 Deployment Diagram

- Hospital System (Hospital Staff Workstation, Inventory Management System)
- Supplier System (Supplier Database, Supplier Portal)
- Cloud Infrastructure (Automated Analytics Engine, Optimization Module, Inventory Database)
- Interactions between Inventory Manager Dashboard, Order Processing System, and Supplier Portal

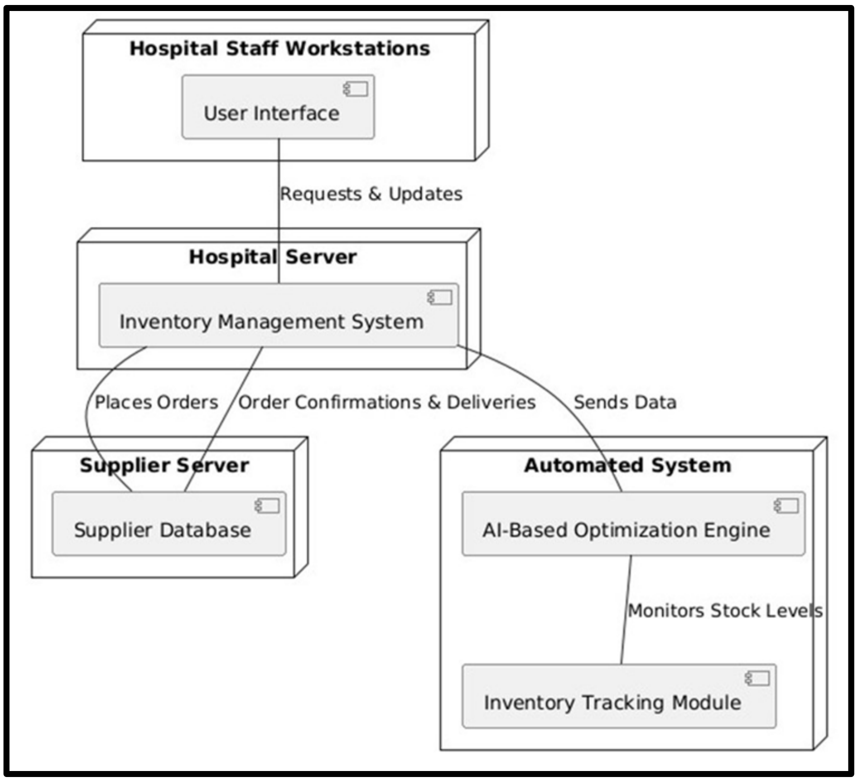

Hospital Staff Workstations
User Interface
Requests & Updates
Hospital Server
Inventory Management System
Places Orders
Order Confirmations & Deliveries
Sends Data
Supplier Server
Supplier Database
Automated System
AI-Based Optimization Engine
Monitors Stock Levels
Inventory Tracking Module

Fig 5.1.5 Deployment Diagram

5.2 FlowChart

A flowchart in UML typically falls under Activity Diagrams, which represent workflows, processes, or system behaviours. In the optimization of medical inventory, a flowchart (UML Activity Diagram) helps visualize how inventory is monitored, forecasted, and replenished efficiently.

Key Elements in the UML Flowchart for Medical Inventory Optimization

1. Start Node → The process begins.
2. Monitor Bounce Rate → Track patient dissatisfaction due to unavailable inventory.
3. Analyze Inventory Shortages → Identify stock levels and predict future demand.
4. Forecast Demand (ML Model: SARIMAX) → Use time-series forecasting to predict stock needs.
5. Stock Categorization (ABC Analysis) → Prioritize inventory based on cost and demand.
6. Optimize Reorder Points & Safety Stock → Set thresholds to avoid shortages.
7. Automated Restocking Alerts → Trigger alerts when inventory is low.
8. Procure Supplies (Cost Optimization) → Purchase within budget constraints.

9. Track Real-Time Inventory & Expiry → Ensure medicines and equipment are used efficiently.
10. Evaluate Performance Metrics → Measure success based on bounce rate reduction, ML accuracy, and revenue growth.
11. End Node → The Process Completes

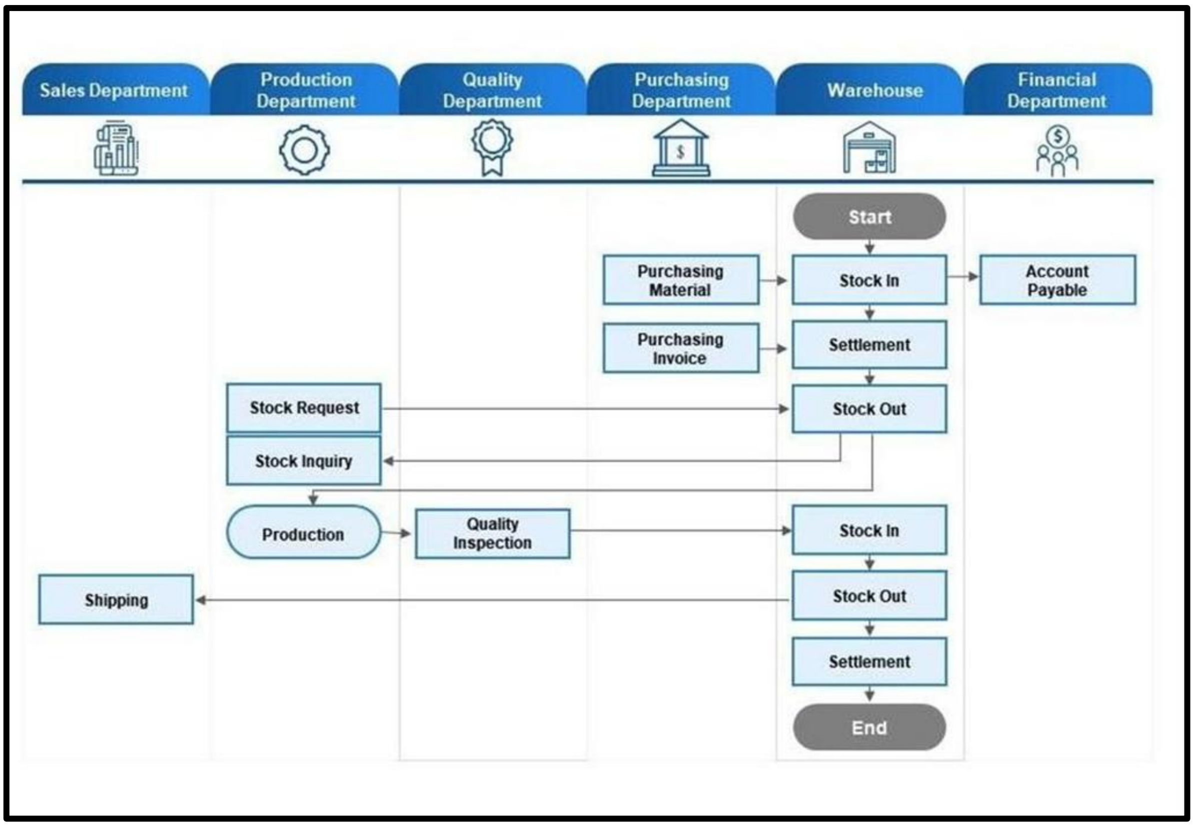


Fig 5.2 FlowChart

CHAPTER 6
MODULES

6.1 MODULES

1. Data Acquisition & Preprocessing Module

Objective: Collect, clean, and preprocess inventory and sales data.

Tasks:

- Extract data from inventory management systems, supplier logs, and patient records.
- Handle missing values, outliers, and data inconsistencies.
- Normalize data for time series forecasting and machine learning models.

Feature engineering:

- Demand seasonality indicators (flu season, emergencies, etc.).
- Supplier lead time analysis.
- Inventory turnover and bounce rate trends.

2. Exploratory Data Analysis (EDA) Module

Objective: Identify patterns in stock usage, demand fluctuations, and bounce rate trends.

Tasks:

- Visualize stock levels, demand trends, and bounce rate over time.
- Conduct ABC-FSN analysis to classify inventory items.
- Compute correlations between stockouts and bounce rates.
- Identify high-impact products causing patient dissatisfaction.

Tools: Python (Pandas, Seaborn, Matplotlib), SQL, Tableau/Power BI

3. Demand Forecasting Module

Objective: Predict inventory demand to reduce stockouts and overstocking.

Tasks:

- Apply time-series forecasting models:
- ARIMA, Prophet (short-term predictions).
- LSTM, XGBoost (long-term predictions).
- Evaluate models using RMSE, MAE, and accuracy (≥90%).
- Implement real-time demand prediction updates.

Tools: Scikit-learn, Statsmodels, TensorFlow/Keras, Prophet

4. Inventory Optimization Module

Objective: Optimize stock levels to minimize costs while ensuring availability.

Tasks:

- Implement Economic Order Quantity (EOQ) and Reorder Point (ROP) models.
- Optimize supplier selection and lead time adjustments.
- Apply Reinforcement Learning (RL) for dynamic inventory adjustments.
- Minimize waste using FIFO/FEFO (First-In-First-Out / First-Expiry-First-Out).

Tools: SciPy (for optimization), OR-Tools, PyTorch (for RL models)

5. Bounce Rate Reduction Module

Objective: Improve inventory tracking to lower patient dissatisfaction.

Tasks:

- Develop a real-time stock monitoring system with RFID/barcode integration.

- ➢ Automate low-stock alerts and predictive ordering.
- ➢ Implement dynamic safety stock levels for high-priority items.

Tools: IoT (RFID/Barcode), Flask/Django (for alerts), SQL databases

6. Cost Optimization & Economic Impact Module

Objective: Ensure financial efficiency and achieve economic success criteria.

Tasks:

- ➢ Analyze holding costs, ordering costs, and waste reduction impact.
- ➢ Simulate cost-saving scenarios with different inventory strategies.
- ➢ Track revenue improvements (target: 20 lacs INR increase).

Tools: Excel (Cost Analysis), Python (Simulation), Power BI/Tableau

7. Dashboard & Reporting Module

Objective: Provide real-time analytics for better decision-making.

Tasks:

- ➢ Build an interactive dashboard to track inventory health.
- ➢ Display forecasted demand, current stock levels, and bounce rates.
- ➢ Implement KPIs monitoring:
    - Stock turnover rate
    - Forecast accuracy
    - Cost savings

Tools: Power BI, Tableau, Streamlit, Flask/Django (for web-based dashboards)

6.2 View & Authorization Users

For user authentication and authorization, you need a User Management Module that ensures secure access control for different stakeholders (e.g., pharmacists, doctors, inventory managers, and admins).

User Management Module

Objective: Implement user authentication, role-based access control (RBAC), and activity tracking.

1. User Authentication (Login & Signup)

Features:

- Secure login/signup system (Email, OTP, or Multi-Factor Authentication).
- Store hashed passwords (bcrypt/Argon2).
- Implement JWT-based authentication for API security.
- OAuth integration (Google, Microsoft, Hospital SSO).

Tools: Flask/Django (Backend), Firebase/Auth0 (Cloud Authentication), PostgreSQL/MySQL (User Database)

2. Role-Based Access Control (RBAC)

Roles & Permissions

Role	**Permissions**
Admin	Full access: Manage users, inventory, and reports
Pharmacist	View stock, update orders, and check expiration
Doctor	View inventory, request medicines
Inventory Manager	Restock inventory, generate reports
Finance Team	Analyze cost reports, approve purchases

Implementation Steps:

- Create user roles in the database.
- Assign permissions based on roles (e.g., Pharmacists can edit stock, Doctors can only view).
- Use middleware to restrict access to APIs/pages based on roles.

Tools: Flask/Django RBAC, Role-Based Middleware (Node.js/Express), Firebase Authentication

3. User Activity Logs & Security

Features:

- Track login history (who accessed what and when).
- Monitor inventory actions (stock updates, approvals).
- Alert system for suspicious activities (e.g., bulk deletion of stock).

Tools: ELK Stack (Elasticsearch for logs), AWS CloudWatch, PostgreSQL Audit Tables

4. User Interface & Dashboard

Features:

- Admin dashboard: Manage users, reset passwords, assign roles.
- User profile page: Change passwords, update details.
- Access logs: Show user activity with timestamps.

Tools: React/Next.js (Frontend), Tailwind CSS, Streamlit (for simple dashboards)

6.3 Remote Users

Remote Users in Medical Inventory Optimization

Remote users play a crucial role in optimizing medical inventory by leveraging digital tools, AIdriven analytics, and cloud-based platforms. Here's a breakdown of key remote users and their contributions:

1.Healthcare Providers (Doctors, Nurses, Pharmacists)

- **Role:** Provide insights into real-time demand for medicines and medical supplies.
- **Tools Used:** Telehealth platforms, EMR (Electronic Medical Records), inventory tracking apps.
- **Key Contributions:**

- Report stockouts or surplus remotely.
- Adjust demand forecasts based on patient needs.
- Use e-prescriptions to optimize pharmacy stock.

2.Supply Chain & Inventory Managers

- **Role:** Monitor inventory levels, track supplier performance, and ensure timely restocking.
- **Tools Used:** Cloud-based inventory systems, AI-driven forecasting models, ERP (Enterprise Resource Planning) software.
- **Key Contributions:**

- Adjust orders based on predictive analytics.
- Identify inefficiencies and minimize overstocking.
- Optimize supplier coordination remotely.

3.Data Scientists & Analysts

- **Role:** Develop ML models for demand forecasting and inventory optimization.
- **Tools Used:** Python, R, Tableau, Power BI, AWS/GCP for cloud computing.
- **Key Contributions:**

- Forecast medical supply demand trends.
- Implement AI/ML models to optimize stock levels.
- Monitor inventory KPIs and bounce rate trends.

4.IT & Software Engineers

- **Role:** Maintain and enhance inventory management systems, automation, and tracking solutions.

- **Tools Used:** RFID tracking, barcode systems, cloud databases, APIs for integration.
- **Key Contributions:**

- Develop automated reordering systems.
- Enhance cybersecurity for inventory data.
- Improve system efficiency through AI-driven tracking.

5.Hospital Administrators & Financial Teams

- **Role:** Oversee cost management, budgeting, and economic success of inventory optimization.
- **Tools Used:** ERP systems, financial dashboards, AI-driven cost analysis tools.
- **Key Contributions:**

- Monitor inventory costs and balance budgets.
- Ensure compliance with procurement policies.
- Optimize stock levels to align with financial goals.

6.Remote Vendors & Logistics Providers

- **Role:** Ensure timely delivery and distribution of medical supplies.
- **Tools Used:** IoT-enabled tracking, VMI (Vendor Managed Inventory), blockchain for supply chain.
- **Key Contributions:**

- Use real-time data to manage inventory logistics.
- Prevent shortages through proactive restocking.
- Minimize waste by optimizing delivery schedules.

CHAPTER 7
TECHNOLOGY DESCRIPTION

7.1 Python Introduction

Python is a high-level programming language that can be used for a variety of purposes. Programming language Python was founded in 1991 by Guido van Rossum and features a design philosophy that prioritises code readability, including the use of wide space. dynamic type systems and memory management are two of Python's features. Object-oriented and imperative programming, as well as functional and procedural approaches, are all supported by the large and extensive library.

Python Scripting Language The interpreter works with Python while it runs. Compiling it before launching a programme is not required. In this respect, it is similar to Perl or PHP.

Modules used in this project:

NumPy

Please note that this is a collection of general-purpose array processing functions. An array object, as well as a collection of utilities for working with them, are provided by the library. In scientific computing, it's the most essential Python package out there. As a starting point, here are a few: An N-dimensional array object having a strong structure. Broadcasting has high-tech capabilities. Tools for integrating C/C++ and Fortran code Capabilities in linear algebra,

Pandas

Using strong data structures, this free Python library is able to handle and analyse high performance data. When it comes to doing calculations, Python was the go-to language. Analysis of data was not affected in any substantial way. How to repair it? Pandas found out.

Matplotlib

The Python 2D charting toolkit Matplotlib's publications look beautiful on every device, whether they're printed on paper or displayed in an interactive environment.

There are a variety of places where Matplotlib can be used: scripts written with either Python or I Python; Jupiter Notebooks; web app servers; and four different GUI frameworks. Matplotlib strives to make things difficult in order to make things easier to understand

Learning using Scikit-Learn

Scikit-learn provides a common Python interface for a variety of learning algorithms, both supervised and unsupervised. There are several Linux distributions available, making it suitable for both academic and commercial use. In addition to being a high-level programming language, Python may be interpreted and utilised for a wide variety of purposes. Guido van Rossum, the man behind Python's 1991 launch, prioritised code readability by inserting a lot of whitespace in the language's design.

7.2 History of Python

What's the link between the alphabet and Python, you might be wondering? This is a true ABC. When we talk about ABC in Python, it's clear that we're referring to the programming language

ABC. ABC is a general-purpose programming language and environment created at the CWI in Amsterdam, the Netherlands (Centrum Wiskunde&Informatica). As early as the late 1980s, the design of ABC had an enormous impact on Python. An operating system developed at CWI by Guido van Rossum, called Amoeba, was in use. At Centrum voorWiskundeenInformatica in the early 1980s, Guido van Rossum was part of a team developing a language dubbed ABC (CWI). The ABC project's influence on Python may be obscure, but I make an effort to acknowledge it because A few examples of ML in action are provided below.

- Emotional profiling
- Emotional analysis......

  For example, the ability to detect and prevent errors
- Prediction and forecasting of the weather
- Analyzing and predicting the stock market
- Synthesis of speech
- Speech synthesis

- Segmentation of customers
- Recognition of objects
- Detection of fraud

7.3 Installation of Python

How to install Python on Windows?

Python is a high-level programming language that has become increasingly popular due to its simplicity, versatility, and extensive range of applications. The process of How to install Python in Windows, operating system is relatively easy and involves a few uncomplicated steps.

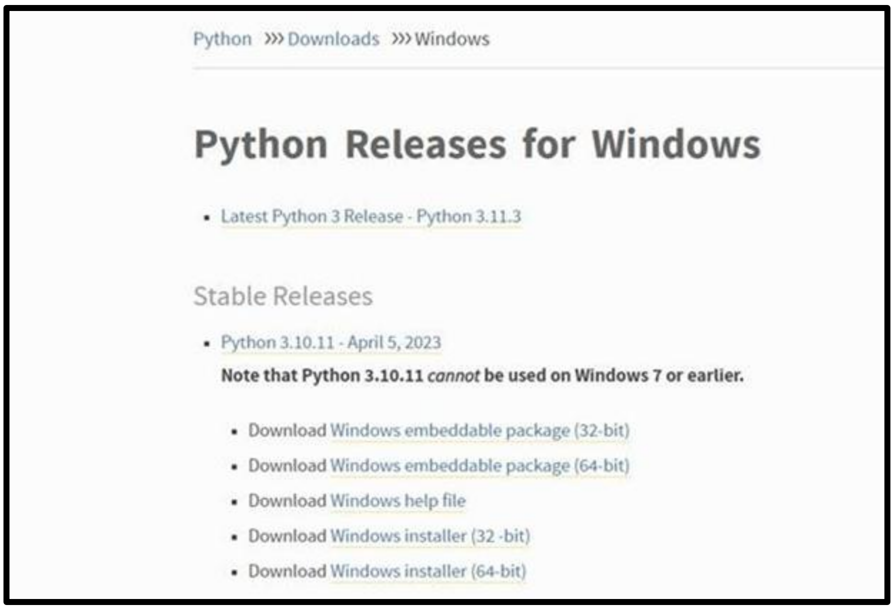
This article aims to take you through the process of downloading and installing Python on your Windows computer. To Install Python on Linux or Mac visit the below link:

How to Install Python in Windows?

We have provided step-by-step instructions to guide you and ensure a successful installation. Whether you are new to programming or have some experience, mastering how to install Python on Windows will enable you to utilize this potent language and uncover its full range of potential applications.To download Python on your system, you can use the following steps

Step 1: Select Version to Install Python

Visit the official page for Python https://www.python.org/downloads/ on the Windows operating system. Locate a reliable version of Python 3, preferably version 3.10.11, which was used in testing this tutorial. Choose the correct link for your device from the options provided: either Windows installer (64-bit) or Windows installer (32-bit) and proceed to download the executable file.



Python >>> Downloads >>> Windows

Python Releases for Windows

- Latest Python 3 Release - Python 3.11.3

Stable Releases

- Python 3.10.11 - April 5, 2023

  Note that Python 3.10.11 *cannot* be used on Windows 7 or earlier.

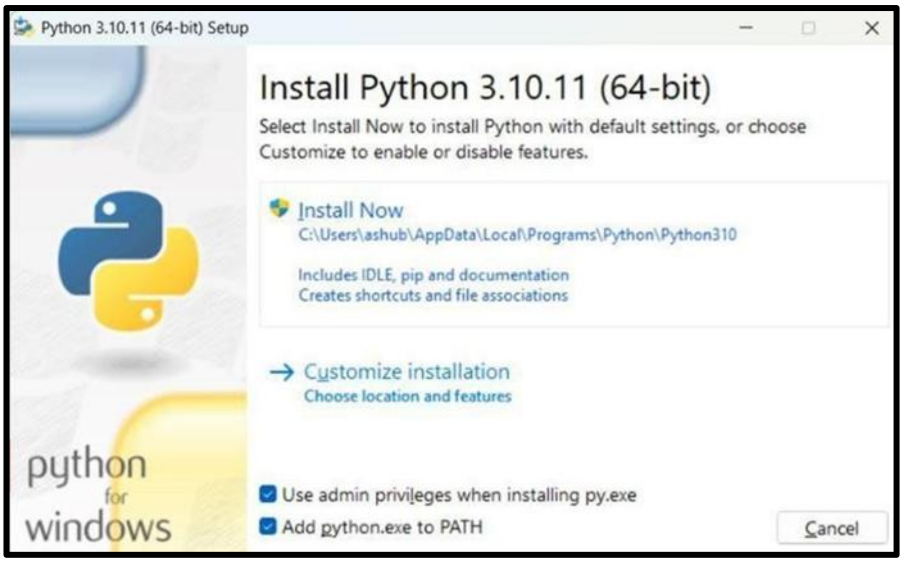
  - Download Windows embeddable package (32-bit)
  - Download Windows embeddable package (64-bit)
  - Download Windows help file
  - Download Windows installer (32 -bit)
  - Download Windows installer (64-bit)

Fig 7.3.1 Versions to Install Python Window

Python Homepage

Step 2: Downloading the Python Installer

Once you have downloaded the installer, open the .exe file, such as python-3.10.11-amd64.exe, by double-clicking it to launch the Python installer. Choose the option to Install the launcher for all users by checking the corresponding checkbox, so that all users of the computer can access the Python launcher application.Enable users to run Python from the command line by checking the Add python.exe to PATH checkbox.



Python 3.10.11 (64-bit) Setup

Install Python 3.10.11 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\ashub\AppData\Local\Programs\Python\Python310

Includes IDLE, pip and documentation
Creates shortcuts and file associations

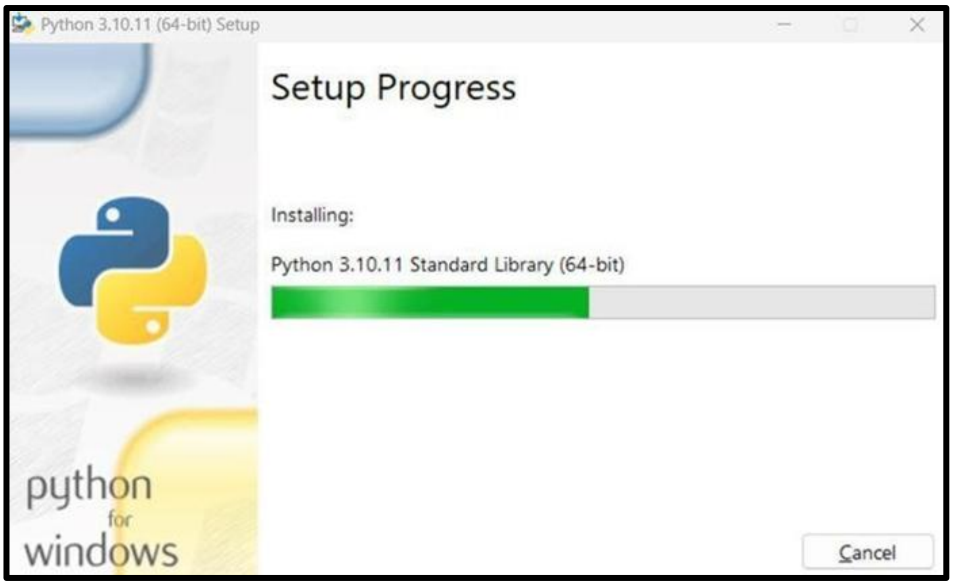
Customize installation
Choose location and features

☑ Use admin privileges when installing py.exe
☑ Add python.exe to PATH

Fig 7.3.2 Downloading the Python Installer Window1

Python Installer

After Clicking the Install Now Button the setup will start installing Python on your Windows system. You will see a window like this.



Setup Progress

Installing:

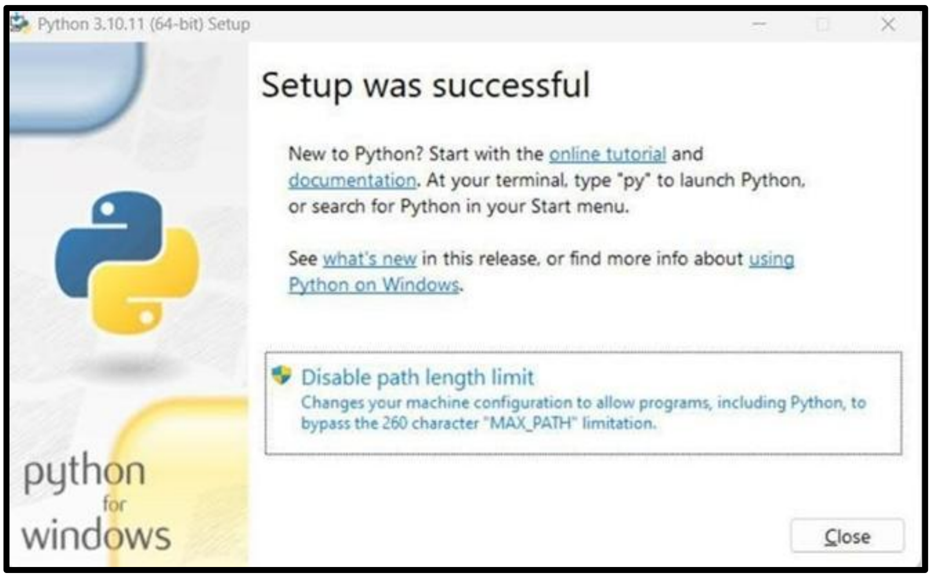
Python 3.10.11 Standard Library (64-bit)

Fig 7.3.2 Downloading the Python Installer Window2

Python Setup

Step 3: Running the Executable Installer

After completing the setup. Python will be installed on your Windows system. You will see a successful message.



Setup was successful

New to Python? Start with the online tutorial and documentation. At your terminal, type "py" to launch Python, or search for Python in your Start menu.

See what's new in this release, or find more info about using Python on Windows.

Disable path length limit

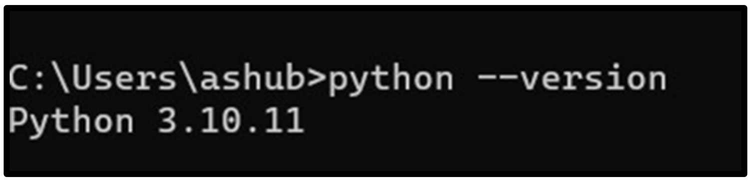
Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.

Fig 7.3.3 Running the Executable Installer Window

Step 4: Verify the Python Installation in Windows

Close the window after successful installation of Python. You can check if the installation of Python was successful by using either the command line or the Integrated Development Environment (IDLE), which you may have installed. To access the command line, click on the Start menu and type "cmd" in the search bar.

Then click on Command Prompt. **python --version**

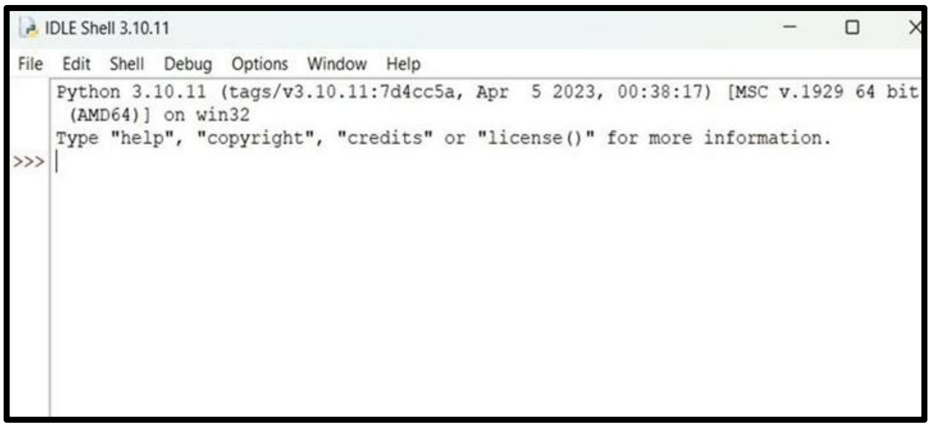


```
C:\Users\ashub>python --version
Python 3.10.11
```

Fig 7.3.4 Verify the Python Installation Window1

Python version

You can also check the version of Python by opening the IDLE application. Go to Start and enter IDLE in the search bar and then click the IDLE app, for example, IDLE (Python 3.10.11 64-bit). If you can see the Python IDLE window then you are successfully able to download and installed Python on Windows.



```
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.1929 64 bit
 (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

Fig 7.3.5 Verify the Python Installation Window

Python IDLE

Getting Started with Python

Python is a lot easier to code and learn. Python programs can be written on any plain text editor like Notepad, notepad++, or anything of that sort. One can also use an

Online IDE to run Python code or can even install one on their system to make it more feasible to write these codes because IDEs provide a lot of features like an intuitive code editor, debugger, compiler, etc. To begin with, writing Python Codes and performing various intriguing and useful operations, one must have Python installed on their System.

7.4 Spyder

Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection and beautiful visualization capabilities of a scientific package.\n Furthermore, Spyder offers built-in integration with many popular scientific packages, including NumPy, SciPy, Pandas, IPython, QtConsole, Matplotlib, SymPy, and more.\n Beyond its many built-in features, Spyder can be extended even further via third-party plugins.\n Spyder can also be used as a PyQt5 extension library, allowing you to build upon its functionality and embed its components, such as the interactive console or advanced editor, in your own software.

Spyder Installation

After installing Anaconda and creating and activating your environment, you can start Spyder on Windows, macOS, or Linux. Launch Spyder using either Navigator or a command line terminal:

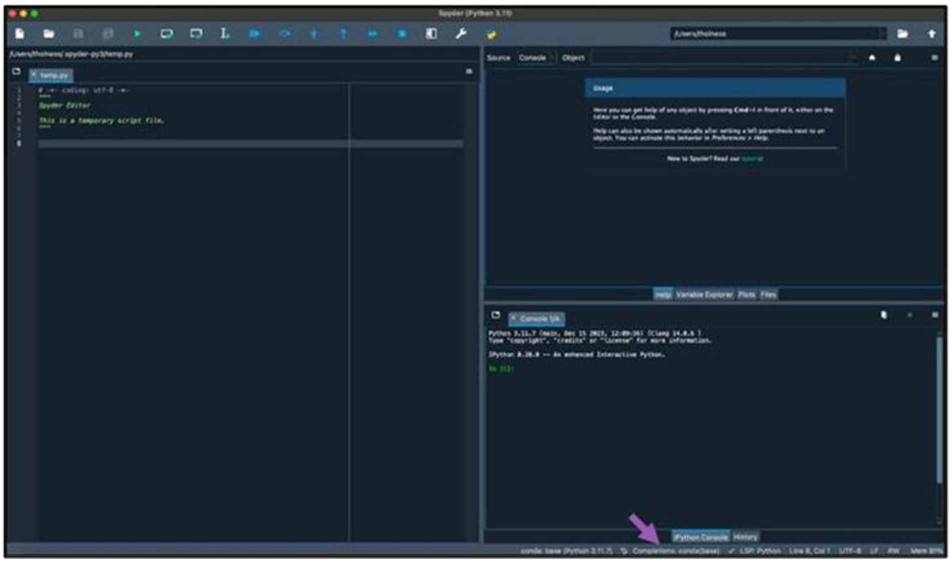
Creating and activating anaconda environment

Before starting your Python project, Anaconda recommends creating a conda environment to isolate your project’s software packages and manage their dependencies. For more information about creating and activating a conda environment, see Environments.

Selecting your environment’s Python interpreter

To ensure your code has access to the correct code completion and type information for your project, you must activate the conda environment associated with your project and select it as the Python interpreter in Spyder.

1.Open Spyder.
2. Select Completions in the Status Bar



```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
```

Here you can get help of any object by pressing Cmd+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.

New to Spyder? Read our tutorial

```
Python 3.11.7 (main, Dec 15 2023, 12:09:56) [Clang 14.0.6 ]
Type "copyright", "credits" or "license" for more information.

IPython 8.20.0 -- An enhanced Interactive Python.

In [1]:
```

```
conda: base (Python 3.11.7)   Completions: conda(base)   ✓ LSP: Python   Line 8, Col 1   UTF-8   LF   RW   Mem 81%
```

Fig 7.4 Spyder Interface Window

Select Use the following Python interpreter.

3.Select your preferred environment.

4.Click Ok.

7.5 Streamlit Process

Streamlit is an open-source app framework in python language. It helps us create beautiful web apps for data science and machine learning in a little time. It is compatible with major python libraries such as scikit-learn, keras, PyTorch, latex, numpy, pandas, matplotlib, etc. Syntax for installing this library is shown below.

Install Streamlit

Before you get started, there're a few things you're going to need :

1.Download an IDE or text editor

2.Install Python in your system (preferably Python 3.7 – Python 3.10)

3.Install PIP in your

system.

Install Anaconda

If you don't have Anaconda install yet, follow this article to install it – how-to-install-anaconda-on windows.

Create a new environment with Streamlit

Now we will be setting up the environment.

- Anaconda provides steps for setting up Anaconda and managing the environment through the Anaconda Navigator. Follow them and setup Anaconda.
- Next, select the " " icon appearing next to your new environment. Then we need to select "Open terminal" from the menu that appears.

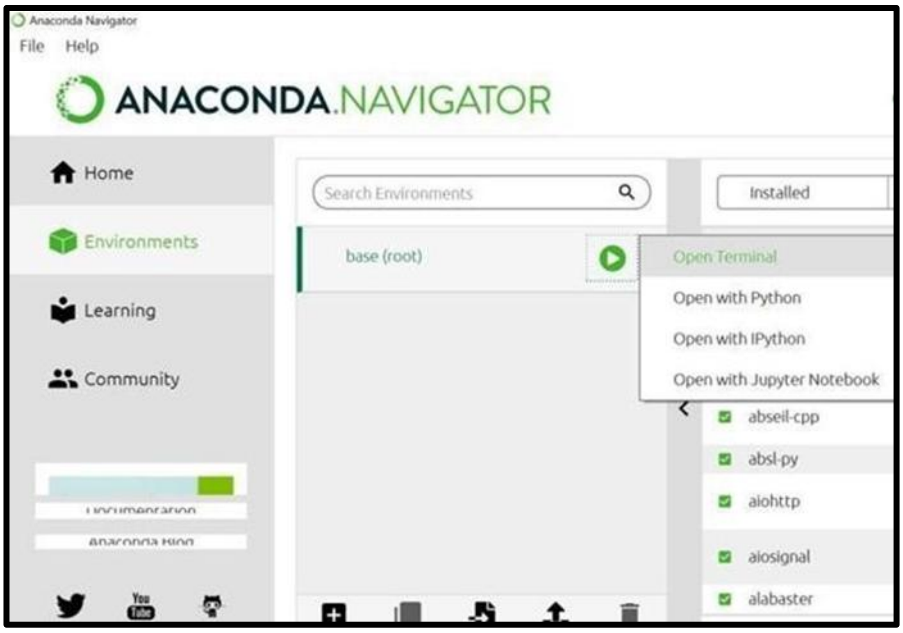
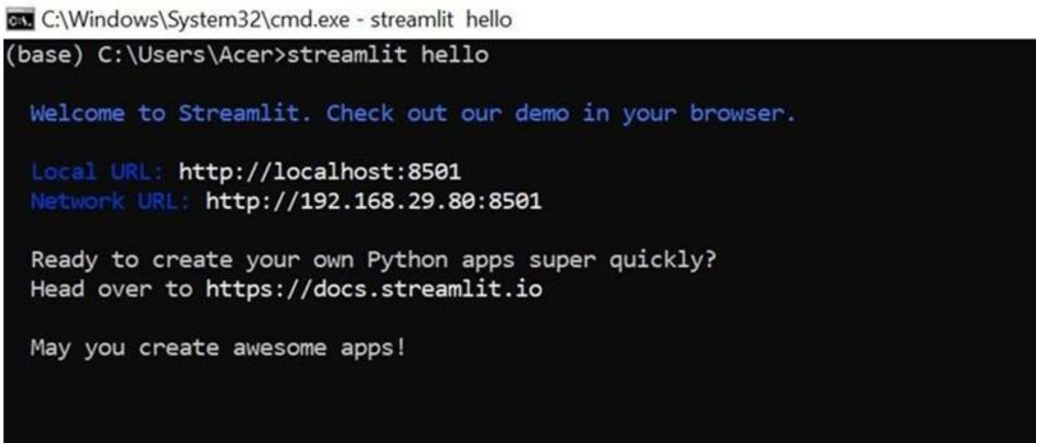


Fig 7.5.1 Anaconda Navigator

Anaconda Navigator

In the command-prompt type

```
pip install streamlit
```



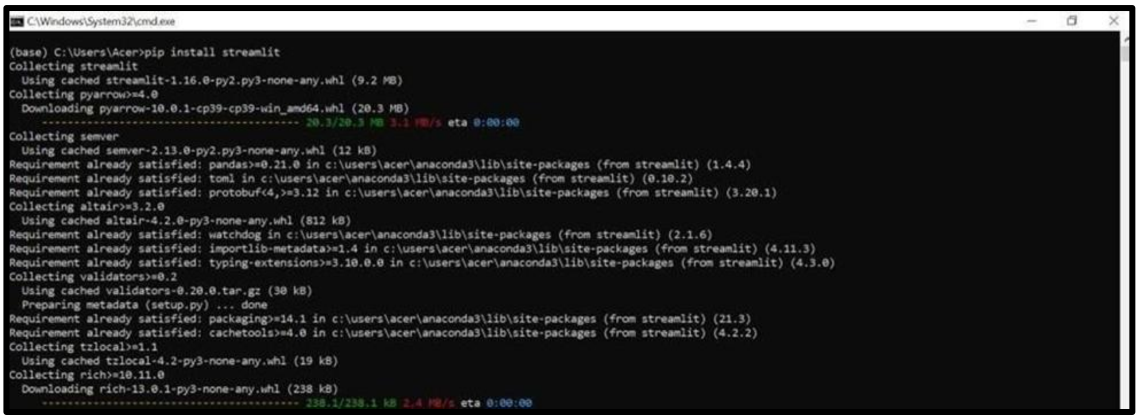
```
C:\Windows\System32\cmd.exe - streamlit  hello
(base) C:\Users\Acer>streamlit hello

  Welcome to Streamlit. Check out our demo in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.29.80:8501

  Ready to create your own Python apps super quickly?
  Head over to https://docs.streamlit.io

  May you create awesome apps!
```

Fig 7.5.2 Anaconda Prompt Window1

Now, test that the installation worked using the command-

```
Streamlit hello
```

Creating a Simple application (Hello World)



```
C:\Windows\System32\cmd.exe

(base) C:\Users\Acer>pip install streamlit
Collecting streamlit
  Using cached streamlit-1.16.0-py2.py3-none-any.whl (9.2 MB)
Collecting pyarrow>=4.0
  Downloading pyarrow-10.0.1-cp39-cp39-win_amd64.whl (20.3 MB)
     ---------------------------------------- 20.3/20.3 MB 3.1 MB/s eta 0:00:00
Collecting semver
  Using cached semver-2.13.0-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: pandas>=0.21.0 in c:\users\acer\anaconda3\lib\site-packages (from streamlit) (1.4.4)
Requirement already satisfied: toml in c:\users\acer\anaconda3\lib\site-packages (from streamlit) (0.10.2)
Requirement already satisfied: protobuf<4,>=3.12 in c:\users\acer\anaconda3\lib\site-packages (from streamlit) (3.20.1)
Collecting altair>=3.2.0
  Using cached altair-4.2.0-py3-none-any.whl (812 kB)
Requirement already satisfied: watchdog in c:\users\acer\anaconda3\lib\site-packages (from streamlit) (2.1.6)
Requirement already satisfied: importlib-metadata>=1.4 in c:\users\acer\anaconda3\lib\site-packages (from streamlit) (4.11.3)
Requirement already satisfied: typing-extensions>=3.10.0.0 in c:\users\acer\anaconda3\lib\site-packages (from streamlit) (4.3.0)
Collecting validators>=0.2
  Using cached validators-0.20.0.tar.gz (30 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: packaging>=14.1 in c:\users\acer\anaconda3\lib\site-packages (from streamlit) (21.3)
Requirement already satisfied: cachetools>=4.0 in c:\users\acer\anaconda3\lib\site-packages (from streamlit) (4.2.2)
Collecting tzlocal>=1.1
  Using cached tzlocal-4.2-py3-none-any.whl (19 kB)
Collecting rich>=10.11.0
  Downloading rich-13.0.1-py3-none-any.whl (238 kB)
     ---------------------------------------- 238.1/238.1 kB 2.4 MB/s eta 0:00:00
```

Fig 7.5.3 Anaconda Prompt Window2

The 'hello, world!' script in Streamlit: streamlit hello

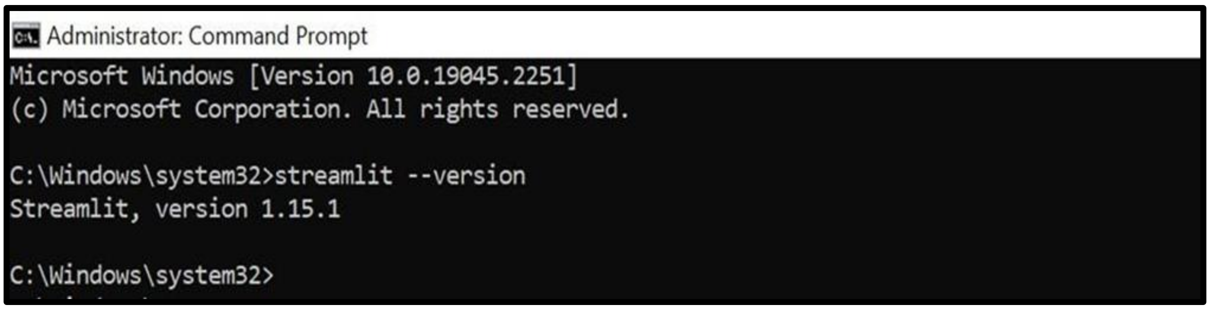
# to run your python script streamlit run myFirstStreamlitApp.py

You can stop running your app any time using Ctrl + C.

Open the terminal and write the command in your terminal:

```
Streamlit --version
```

This command should tell you what version the installed streamlit package is.



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19045.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>streamlit --version
Streamlit, version 1.15.1

C:\Windows\system32>
```

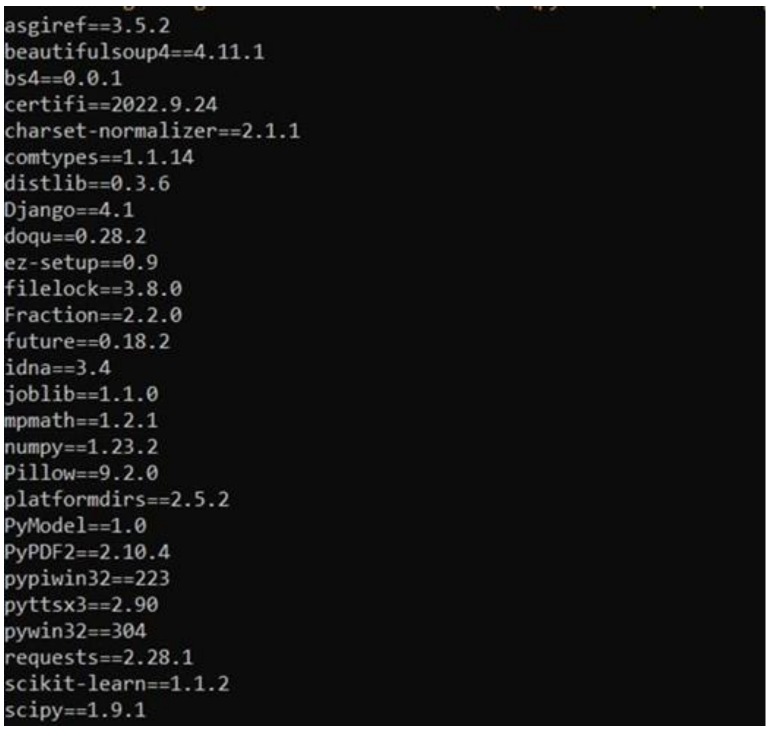
Fig 7.5.4 Anaconda Prompt Window

Check the Streamlit Version using the pip

Using the pip command gives all the detail of current versions of python modules which you installed in your system.

To check Streamlit version,open your terminal and write the pip command in your terminal. If Streamlit is not currently installed in your system, it will not appear in the list of modules.

```
pip freeze
```



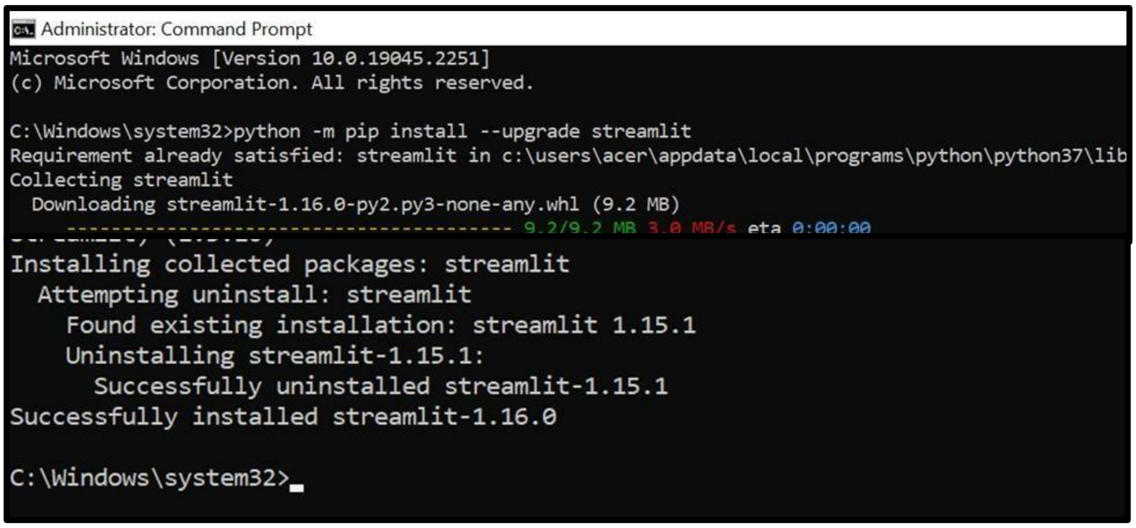
```
asgiref==3.5.2
beautifulsoup4==4.11.1
bs4==0.0.1
certifi==2022.9.24
charset-normalizer==2.1.1
comtypes==1.1.14
distlib==0.3.6
Django==4.1
doqu==0.28.2
ez-setup==0.9
filelock==3.8.0
Fraction==2.2.0
future==0.18.2
idna==3.4
joblib==1.1.0
mpmath==1.2.1
numpy==1.23.2
Pillow==9.2.0
platformdirs==2.5.2
PyModel==1.0
PyPDF2==2.10.4
pypiwin32==223
pyttsx3==2.90
pywin32==304
requests==2.28.1
scikit-learn==1.1.2
scipy==1.9.1
```

Fig 7.5.5 Anaconda Prompt Window4

Update Streamlit Version

If you want to update your Streamlit version you can use any one of these commands to update your Streamlit.

```
Python -m pip install –upgrade Streamlit
Python -m pip install -U Streamlit
```



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19045.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>python -m pip install --upgrade streamlit
Requirement already satisfied: streamlit in c:\users\acer\appdata\local\programs\python\python37\lib
Collecting streamlit
  Downloading streamlit-1.16.0-py2.py3-none-any.whl (9.2 MB)
     ------------------------------------ 9.2/9.2 MB 3.0 MB/s eta 0:00:00
Installing collected packages: streamlit
  Attempting uninstall: streamlit
    Found existing installation: streamlit 1.15.1
    Uninstalling streamlit-1.15.1:
      Successfully uninstalled streamlit-1.15.1
Successfully installed streamlit-1.16.0

C:\Windows\system32>_
```

Fig 7.5.6 Anaconda Prompt Window5

Advantages of Streamlit

1.It embraces python-scripting.

2.Less code is needed to create amazing web-apps.

3.No callbacks are needed since widgets are treated as variables.

4.Data caching simplifies and speeds up computation pipelines.

CHAPTER 8
SYSTEM TESTING

System Testing is a black-box testing technique where the entire system is tested as a whole. It ensures that all integrated components work correctly and meet business requirements before deployment.

Objectives of System Testing

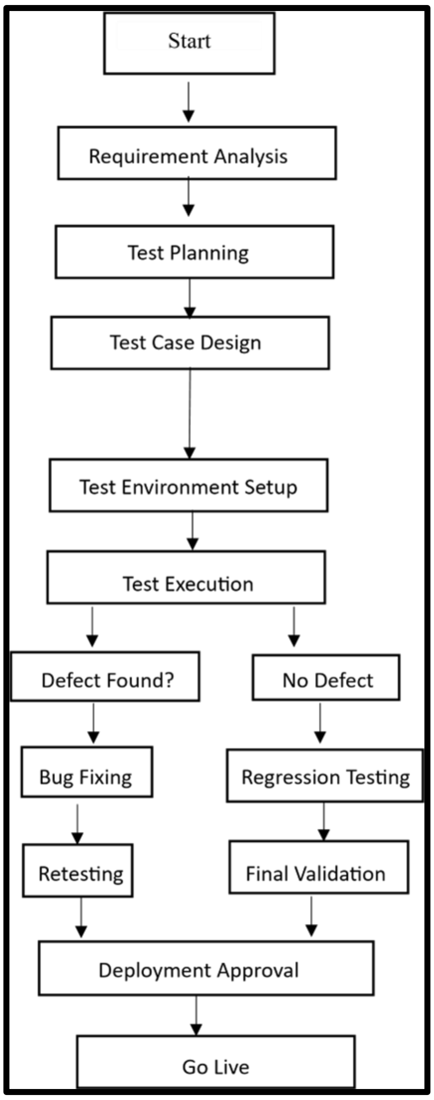
- Verify functionality – Ensure the system meets business needs
- Identify defects – Find and fix bugs before release
- Validate system behavior – Test different workflows and scenarios
- Ensure performance & security – Check response times and vulnerabilities
- Confirm integration – Ensure all modules communicate correctly

8.1 Types of System Testing

- Functional Testing – Ensures accurate stock updates
- Performance Testing – Prevents system slowdowns during high usage
- Security Testing – Protects patient and inventory data
- Usability Testing – Ensures smooth workflow for hospital staff
- Database Testing – Ensures accurate and reliable inventory data

8.1.1 Unit Testing

Unit testing is a software testing method where individual components (or "units") of a system are tested independently to ensure they function correctly. It is typically done by developers during the early stages of development.



8.1.2 Integration Testing

Integration Testing is a software testing method where individual modules (that have already passed unit testing) are combined and tested as a group to ensure they work together correctly. It helps identify data flow issues, interface mismatches, and interaction errors between modules.

Why is Integration Testing Important?

- Ensures seamless communication between components
- Detects defects in module interactions early
- Prevents data loss and inconsistency
- Verifies real-world workflow execution

8.1.3 Functional Testing

Functional Testing is a black-box testing technique that verifies whether a system's functions work as expected based on business requirements. It focuses on what the system does, rather than how it does it.

- ➢ Understand Requirements – Identify functional specifications
- ➢ Design Test Cases – Define test scenarios with input and expected output
- ➢ Set Up Test Environment – Prepare test data and system configuration
- ➢ Execute Tests – Run tests manually or using automation tools
- ➢ Analyze Results – Compare actual vs. expected outputs
- ➢ Report & Fix Bugs – Log and resolve issues
- ➢ Retest & Validate – Ensure fixes work correctly
- ➢ **Example:** Functional Testing in Medical Inventory Optimization

For a medical inventory system, functional testing should cover:

- ➢ Stock Management – Verify correct stock updates after dispensing medicines
- ➢ Reorder Functionality – Ensure automatic reordering works for low stock
- ➢ User Authentication – Test login and access controls for hospital staff
- ➢ Supplier Integration – Validate order placement and confirmation workflows

8.1.4White Box Testing

White Box Testing (also known as Glass Box Testing, Clear Box Testing, or Structural Testing) is a software testing technique that examines the internal code structure, logic, and algorithms of an application. Unlike black-box testing, which focuses on functionality, white-box testing requires knowledge of the code and is typically performed by developers. White Box Testing Process

- ➢ Understand the Code Structure – Analyze the program's logic, functions, and flow Identify Testable Paths – Determine conditions, loops, and decision points

- Design Test Cases – Cover different execution paths, edge cases, and error handling Execute Tests – Run unit tests, integration tests, or static code analysis
- Analyze Results & Debug – Fix defects and optimize performance
- Re-test & Validate – Ensure all paths are properly tested

White Box Testing in Medical Inventory Optimization

For a medical inventory system, white box testing should cover:

- Stock update logic – Ensure correct stock calculations
- API security testing – Prevent unauthorized access to stock data
- Performance optimization – Detect redundant or slow code
- Error handling – Ensure proper logging and recovery

8.1.5 Black Box Testing

Black Box Testing is a software testing technique that evaluates a system's functionality without knowing its internal code, logic, or structure. It focuses on what the system does rather than how it does it.

Black Box Testing Process

- Understand Requirements – Analyze functional specifications
- Design Test Cases – Define input conditions and expected outputs
- Execute Tests – Run test cases manually or using automated tools
- Compare Actual vs Expected Results – Identify defects
- Report & Fix Bugs – Work with developers to resolve issues
- Retest & Validate – Ensure fixes work correctly Black Box Testing in Medical Inventory Optimization

For a medical inventory system, black box testing should cover:

- Stock Management – Ensure correct inventory updates after dispensing medicines
- Order Processing – Verify reorder triggers when stock is low
- User Authentication – Check login functionality for hospital staff

- Billing & Reporting – Validate invoice generation and stock deductions

8.2 Testing Methodologies

Software Testing Methodologies are strategies and approaches used to test software applications to ensure quality, functionality, security, and performance before deployment. These methodologies help identify defects early, reduce risks, and improve software reliability.

Testing in medical inventory optimization ensures that the system is reliable, efficient, and secure while minimizing stockouts and waste. Below are the essential testing methodologies tailored for a medical inventory management system.

Criteria	**Best Testing Methodology**
Small Functions (Stock Update, Reorder Logic)	Unit Testing
Integration Between Inventory & Billing Systems	Integration Testing
Overall System Functionality	System Testing
Validation by End Users (Pharmacists, Admins)	UAT
Check for Broken Features After Updates	Regression Testing
Check Speed & Scalability	Performance Testing
Identify Security Risks	Security Testing

8.3 Other Testing Methodologies

In addition to standard unit, integration, system, regression, and security testing, medical inventory optimization requires specialized testing methodologies to ensure accuracy, efficiency, and compliance with medical standards. Below are additional advanced testing methodologies that can be applied.

1.Risk-Based Testing (RBT)

- Prioritizes testing based on the likelihood and impact of failure.
- Ensures critical functionalities like medicine tracking and stock management are rigorously tested.

Example: Testing low-stock alerts to avoid medicine shortages.

2.Compliance Testing

- Ensures the system complies with medical regulations (HIPAA, FDA, ISO 13485).

Example: Ensuring patient records linked to inventory follow data privacy laws.

3.Data Validation Testing

- Ensures accurate inventory data tracking, updates, and reporting.

Example: Checking expired medicine removal from stock.

4.Automation Testing

- Reduces manual testing effort for repetitive tasks.

Example: Automating medicine search and stock updates using Selenium

5.Usability Testing

- Ensures the inventory system is user-friendly for pharmacists and hospital staff

Example: Checking if stock reports are easy to read and download

CHAPTER 9
PROJECT SCREENSHOTS

9.1 Service Provider

Optimizing medical inventory is crucial for healthcare providers to ensure cost-efficiency, reduce waste, and maintain adequate stock levels for patient care. Here are some service providers and solution providers that specialize in medical inventory optimization:

1. Consulting & Supply Chain Optimization Firms

These firms offer advisory and implementation services:

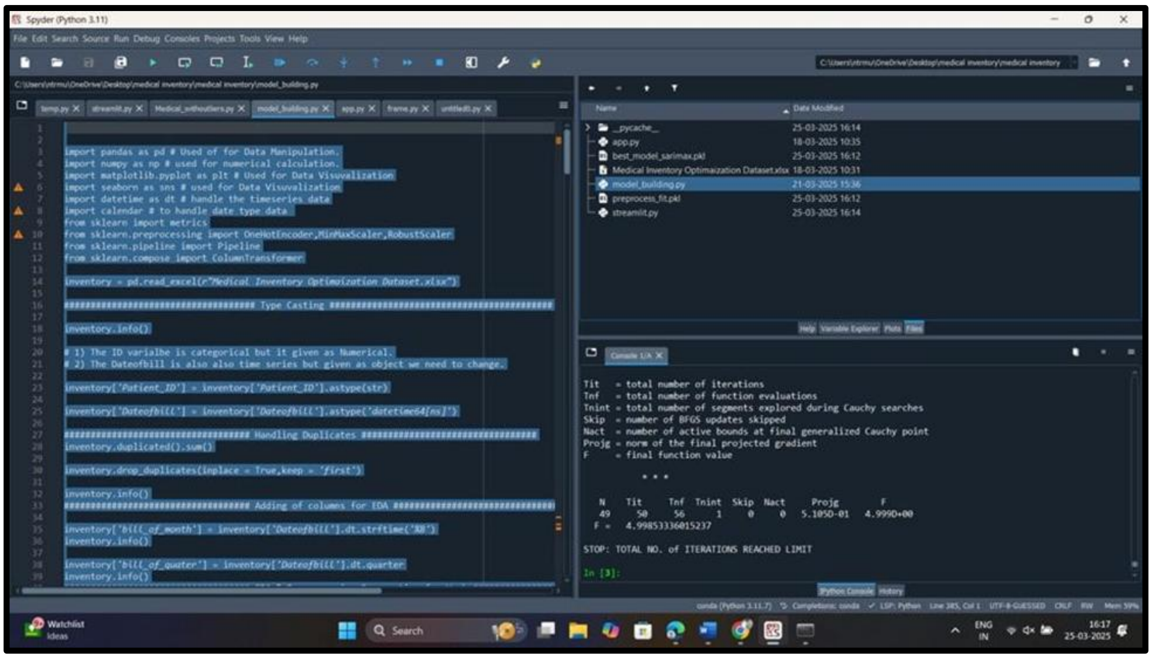
- McKinsey & Company (Healthcare Practice) – Strategic inventory optimization consulting.
- Deloitte Healthcare Supply Chain Services – End-to-end inventory and procurement optimization.
- PwC Health Industries – Helps hospitals optimize inventory and reduce costs.

2. Medical Supply Chain & Procurement Companies

- Cardinal Health Inventory Management Solutions – Offers vendor-managed inventory (VMI) solutions.
- Owens & Minor – Provides inventory management and logistics for medical supplies.
- Medline Industries – Inventory optimization services for hospitals and clinics.
    - Automated Replenishment – Just-in-time (JIT) inventory systems.
    - Expiry Management – Reduces waste of perishable medical supplies.
    - Integration with EHR/ERP – Seamless data flow between systems.

9.2 Program ScreenShots

Withoutliers Program

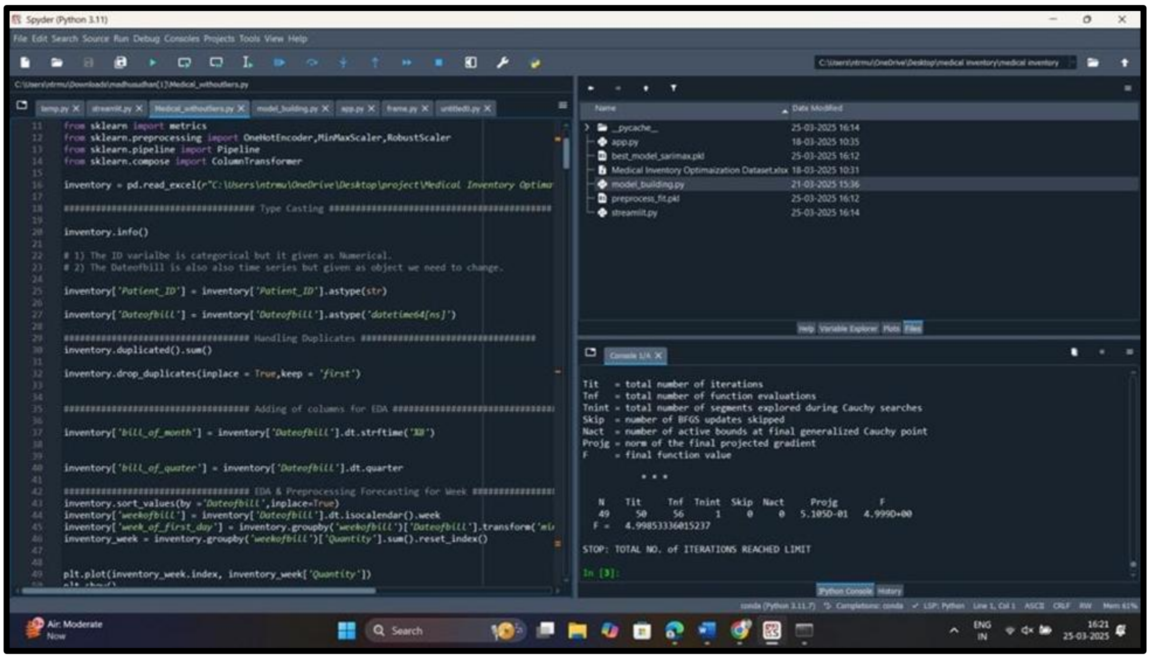


```
import pandas as pd # Used of for Data Manipulation.
import numpy as np # used for numerical calculation.
import matplotlib.pyplot as plt # Used for Data Visuvalization
import seaborn as sns # used for Data Visuvalization
import datetime as dt # handle the timeseries data
import calendar # to handle date type data
from sklearn import metrics
from sklearn.preprocessing import OneHotEncoder,MinMaxScaler,RobustScaler
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

inventory = pd.read_excel(r"Medical Inventory Optimaization Dataset.xlsx")

#################################### Type Casting ##########################################

inventory.info()

# 1) The ID varialbe is categorical but it given as Numerical.
# 2) The Dateofbill is also also time series but given as object we need to change.

inventory['Patient_ID'] = inventory['Patient_ID'].astype(str)

inventory['Dateofbill'] = inventory['Dateofbill'].astype('datetime64[ns]')

#################################### Handling Duplicates ##################################
inventory.duplicated().sum()

inventory.drop_duplicates(inplace = True,keep = 'first')

inventory.info()
#################################### Adding of columns for EDA ############################

inventory['bill_of_month'] = inventory['Dateofbill'].dt.strftime('%B')
inventory.info()

inventory['bill_of_quater'] = inventory['Dateofbill'].dt.quarter
inventory.info()
```

```
Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
   49     50     56      1     0     0   5.105D-01   4.999D+00
  F =   4.99853336015237

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT

In [3]:
```

Fig 9.2.1 Withoutliers Program Window1

Model Building Program

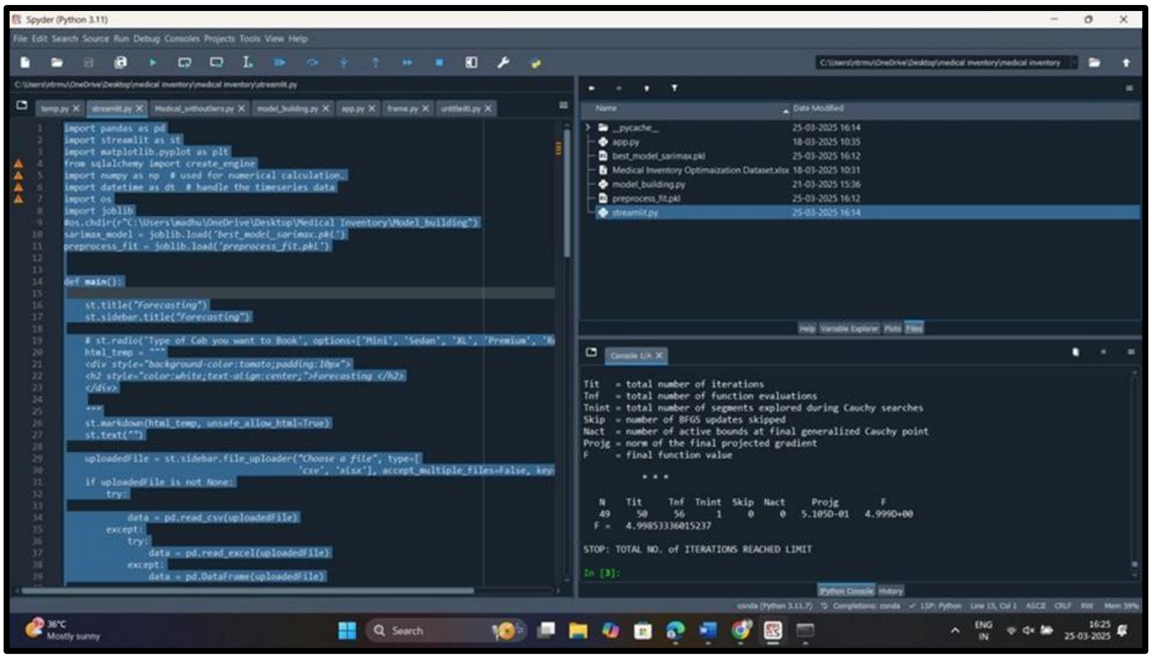


```
from sklearn import metrics
from sklearn.preprocessing import OneHotEncoder,MinMaxScaler,RobustScaler
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

inventory = pd.read_excel(r"C:\Users\ntrmu\OneDrive\Desktop\project\Medical Inventory Optima

#################################### Type Casting ##########################################

inventory.info()

# 1) The ID varialbe is categorical but it given as Numerical.
# 2) The Dateofbill is also also time series but given as object we need to change.

inventory['Patient_ID'] = inventory['Patient_ID'].astype(str)

inventory['Dateofbill'] = inventory['Dateofbill'].astype('datetime64[ns]')

#################################### Handling Duplicates ##################################
inventory.duplicated().sum()

inventory.drop_duplicates(inplace = True,keep = 'first')


#################################### Adding of columns for EDA ############################

inventory['bill_of_month'] = inventory['Dateofbill'].dt.strftime('%B')


inventory['bill_of_quater'] = inventory['Dateofbill'].dt.quarter

#################################### EDA & Preprocessing Forecasting for Week ##############
inventory.sort_values(by ='Dateofbill',inplace=True)
inventory['weekofbill'] = inventory['Dateofbill'].dt.isocalendar().week
inventory['week_of_first_day'] = inventory.groupby('weekofbill')['Dateofbill'].transform('mi
inventory_week = inventory.groupby('weekofbill')['Quantity'].sum().reset_index()


plt.plot(inventory_week.index, inventory_week['Quantity'])
```

```
Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
   49     50     56      1     0     0   5.105D-01   4.999D+00
  F =   4.99853336015237

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT

In [3]:
```

Fig 9.2.2 Model Building Program Window2

Streamlit Program

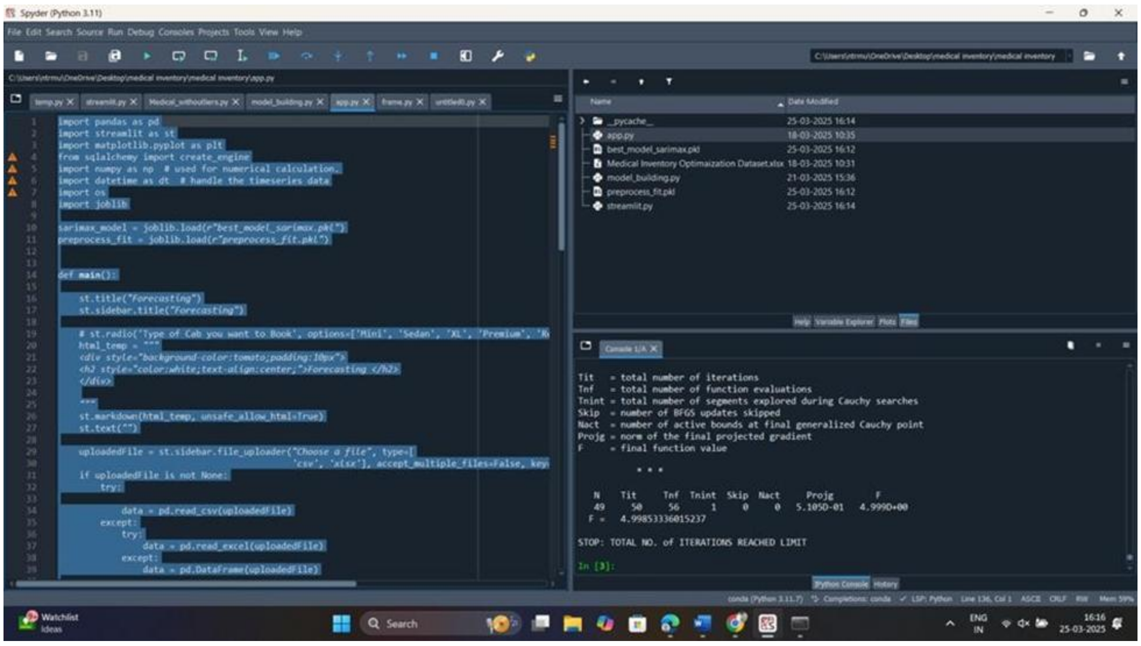


```
import pandas as pd
import streamlit as st
import matplotlib.pyplot as plt
from sqlalchemy import create_engine
import numpy as np  # used for numerical calculation.
import datetime as dt  # handle the timeseries data
import os
import joblib
#os.chdir(r"C:\Users\madhu\OneDrive\Desktop\Medical Inventory\Model_building")
sarimax_model = joblib.load('best_model_sarimax.pkl')
preprocess_fit = joblib.load('preprocess_fit.pkl')


def main():

    st.title("Forecasting")
    st.sidebar.title("Forecasting")

    # st.radio('Type of Cab you want to Book', options=['Mini', 'Sedan', 'XL', 'Premium', 'R
    html_temp = """
    <div style="background-color:tomato;padding:10px">
    <h2 style="color:white;text-align:center;">Forecasting </h2>
    </div>

    """
    st.markdown(html_temp, unsafe_allow_html=True)
    st.text("")

    uploadedFile = st.sidebar.file_uploader("Choose a file", type=[
                                            'csv', 'xlsx'], accept_multiple_files=False, key
    if uploadedFile is not None:
        try:

            data = pd.read_csv(uploadedFile)
        except:
            try:
                data = pd.read_excel(uploadedFile)
            except:
                data = pd.DataFrame(uploadedFile)
```

```
Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
   49     50     56      1     0     0   5.105D-01   4.999D+00
  F =   4.99853336015237

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT

In [3]:
```

Fig 9.2.3 Streamlit Program Window3

App Program (Forecasting)



```
import pandas as pd
import streamlit as st
import matplotlib.pyplot as plt
from sqlalchemy import create_engine
import numpy as np  # used for numerical calculation.
import datetime as dt  # handle the timeseries data
import os
import joblib

sarimax_model = joblib.load(r"best_model_sarimax.pkl")
preprocess_fit = joblib.load(r"preprocess_fit.pkl")


def main():

    st.title("Forecasting")
    st.sidebar.title("Forecasting")

    # st.radio('Type of Cab you want to Book', options=['Mini', 'Sedan', 'XL', 'Premium', 'R
    html_temp = """
    <div style="background-color:tomato;padding:10px">
    <h2 style="color:white;text-align:center;">Forecasting </h2>
    </div>

    """
    st.markdown(html_temp, unsafe_allow_html=True)
    st.text("")

    uploadedFile = st.sidebar.file_uploader("Choose a file", type=[
                                            'csv', 'xlsx'], accept_multiple_files=False, key
    if uploadedFile is not None:
        try:

            data = pd.read_csv(uploadedFile)
        except:
            try:
                data = pd.read_excel(uploadedFile)
            except:
                data = pd.DataFrame(uploadedFile)
```

```
Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
   49     50     56      1     0     0   5.105D-01   4.999D+00
  F =   4.99853336015237

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT

In [3]:
```

Fig 9.2.4 App Program (Forecating) Window4

9.3 Output Screenshots

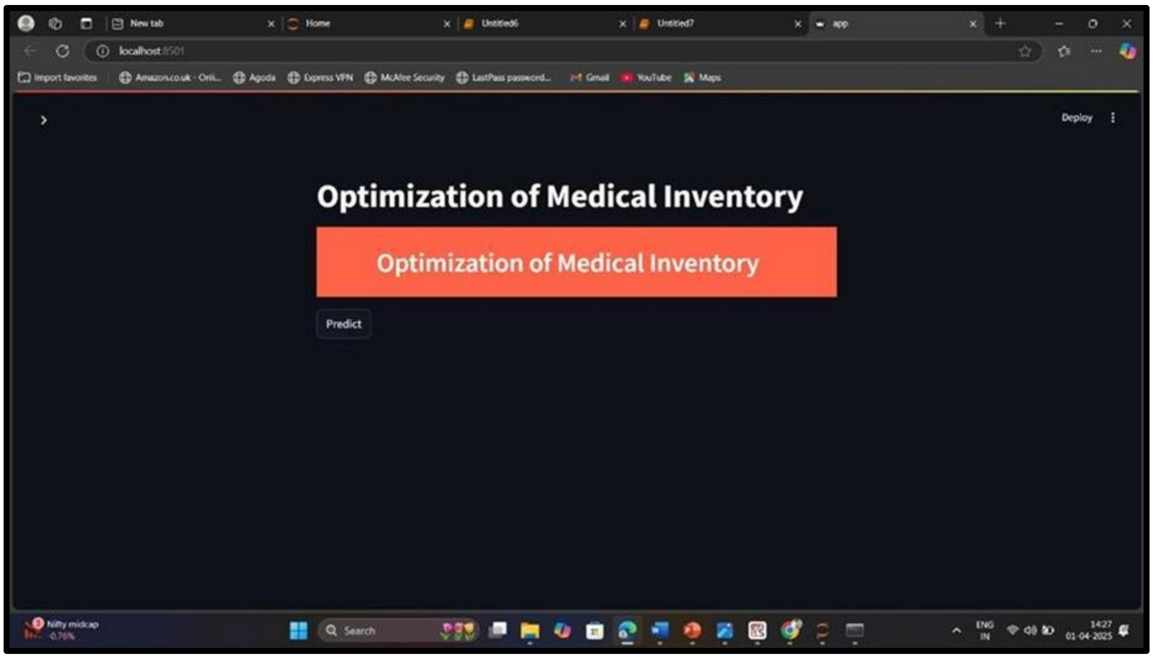
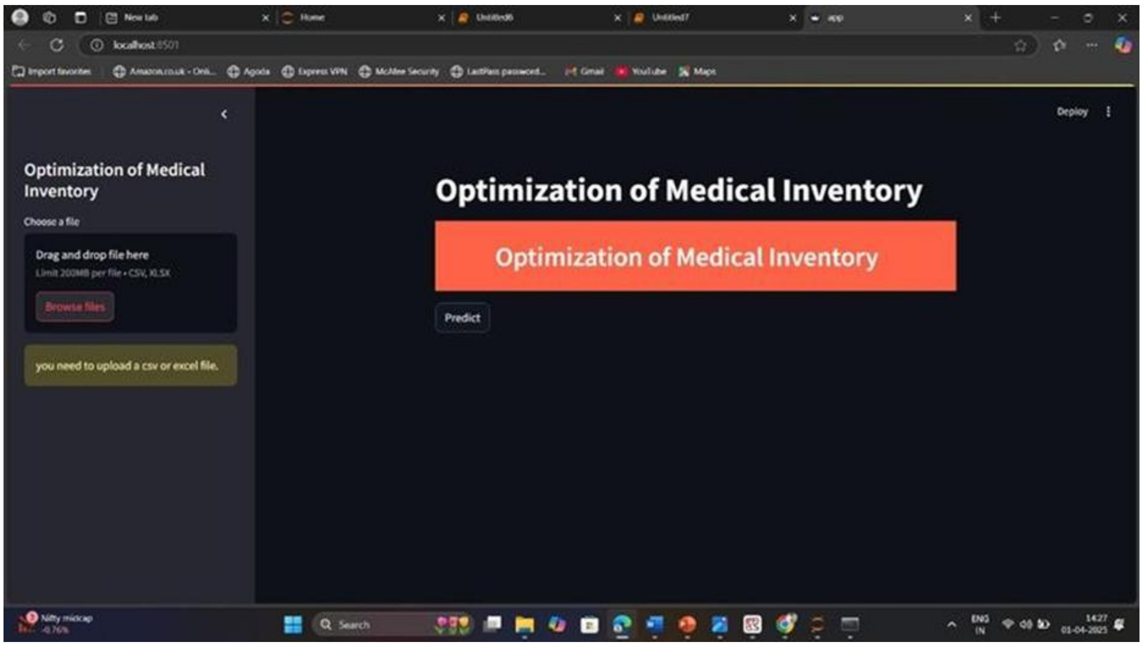

localhost:8501
Deploy
Optimization of Medical Inventory
Optimization of Medical Inventory
Predict

Fig 9.3.1 Optimization of Medical Inventory Window1


localhost:8501
Deploy
Optimization of Medical Inventory
Choose a file
Drag and drop file here
Limit 200MB per file • CSV, XLSX
Browse files
Optimization of Medical Inventory
Optimization of Medical Inventory
Predict

you need to upload a csv or excel file.

Fig 9.3.2 Optimization of Medical Inventory Window2

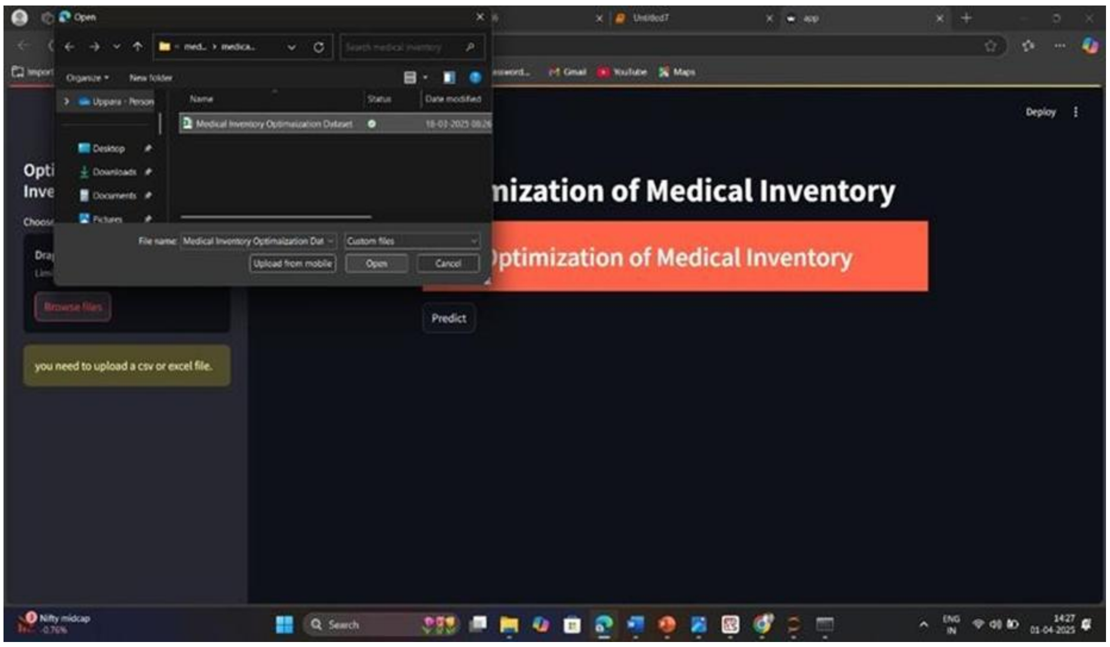

Open
File name: Medical Inventory Optimization Dat
Custom files
Upload from mobile
Open
Cancel
Medical Inventory Optimization Dataset
Optimization of Medical Inventory
Optimization of Medical Inventory
Predict
Browse files
you need to upload a csv or excel file.
Deploy

Fig 9.3.3 Optimization of Medical Inventory Window3

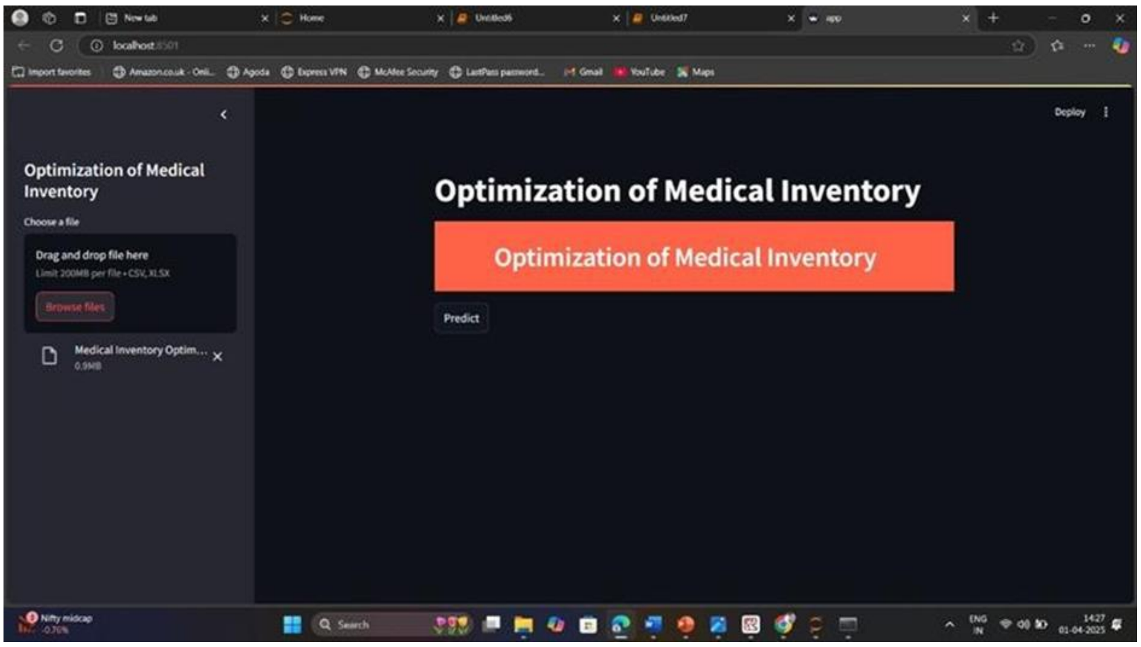
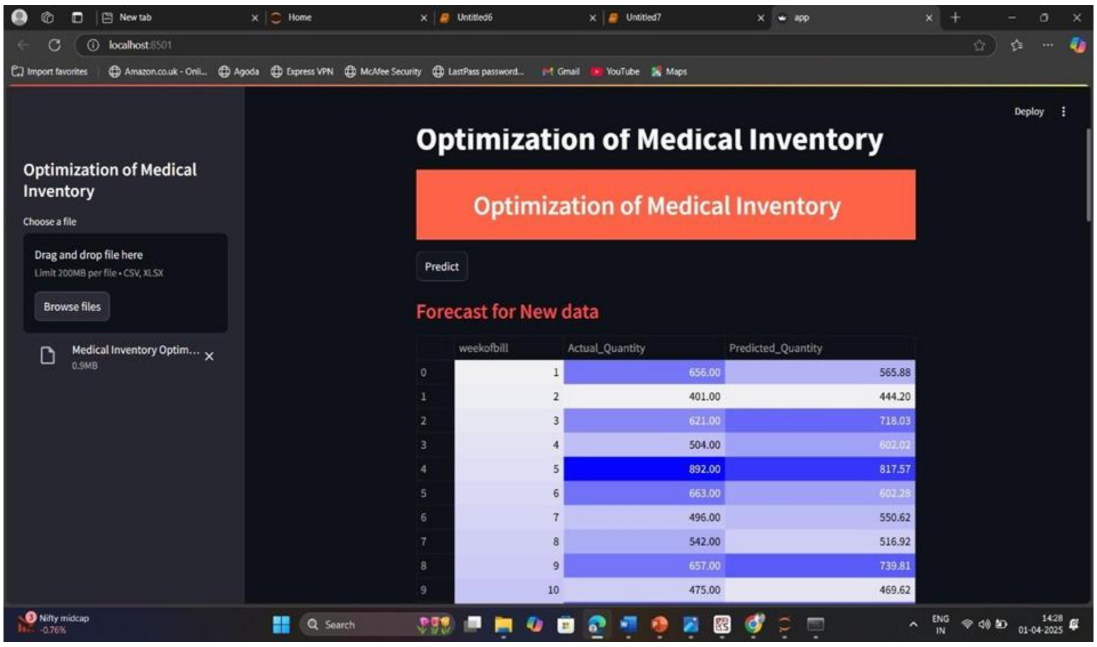

localhost:8501
Optimization of Medical Inventory
Choose a file
Drag and drop file here
Limit 200MB per file • CSV, XLSX
Browse files
Medical Inventory Optim...
0.9MB
Optimization of Medical Inventory
Optimization of Medical Inventory
Predict
Deploy

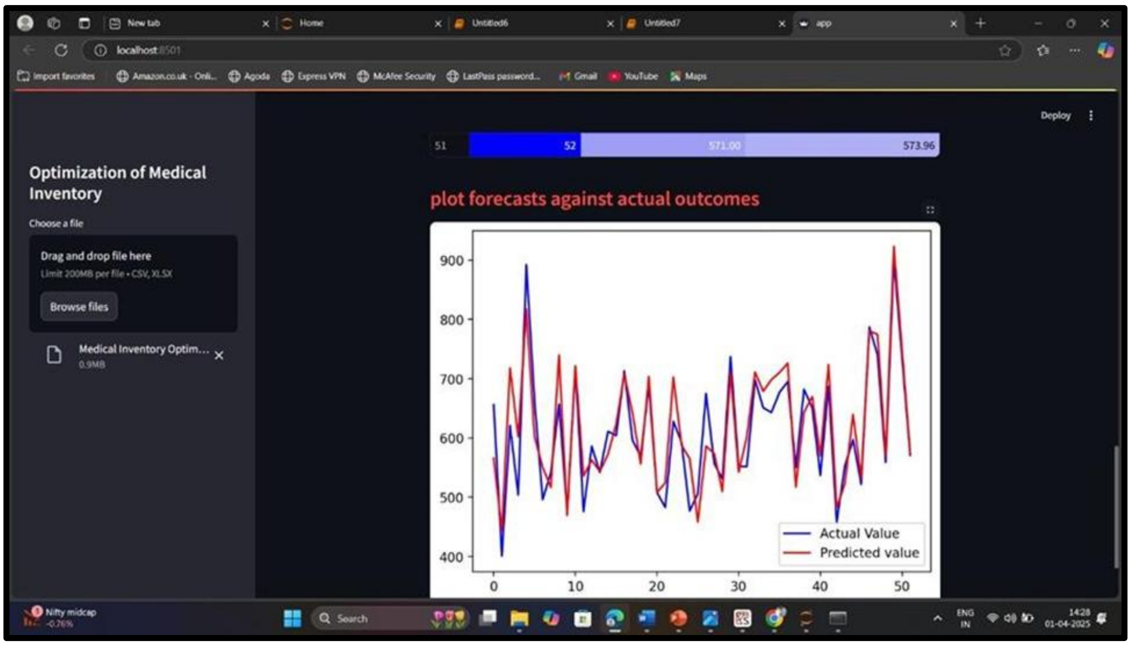
Fig 9.3.4 Optimization of Medical Inventory Window4


Optimization of Medical Inventory
Choose a file
Drag and drop file here
Limit 200MB per file • CSV, XLSX
Browse files
Medical Inventory Optim...
0.9MB
Deploy
Optimization of Medical Inventory
Optimization of Medical Inventory
Predict

Forecast for New data

	weekofbill	Actual_Quantity	Predicted_Quantity
0	1	656.00	565.88
1	2	401.00	444.20
2	3	621.00	718.03
3	4	504.00	602.02
4	5	892.00	817.57
5	6	663.00	602.28
6	7	496.00	550.62
7	8	542.00	516.92
8	9	657.00	739.81
9	10	475.00	469.62

Fig 9.3.5 Optimization of Medical Inventory Window5


Optimization of Medical Inventory
Choose a file
Drag and drop file here
Limit 200MB per file • CSV, XLSX
Browse files
Medical Inventory Optim...
0.9MB
Deploy
plot forecasts against actual outcomes
900
800
700
600
500
400
0
10
20
30
40
50
Actual Value
Predicted value

51	52	571.00	573.96

Fig 9.3.6 Optimization of Medical Inventory Window6

CHAPTER 10
CONCLUSION

The optimization of medical inventory is a critical component of efficient healthcare management. It ensures the timely availability of medical supplies while minimizing waste, reducing costs, and improving patient care. Effective inventory management in the healthcare industry requires a strategic blend of demand forecasting, technology integration, supplier coordination, and regulatory compliance. As healthcare institutions strive to enhance efficiency and reduce operational costs, optimizing medical inventory has emerged as a key priority.

Key Findings

Through various strategies such as demand forecasting, classification techniques (ABC and VED analysis), automation, and real-time tracking, healthcare facilities can significantly improve their inventory management systems. The adoption of Just-in-Time (JIT) inventory models helps reduce storage costs, while AI-driven forecasting minimizes the risk of stockouts and overstocking. Additionally, automation through RFID, IoT, and integrated inventory management systems ensures real-time monitoring, reducing human errors and improving accuracy.Effective supplier relationship management has also proven to be a crucial factor in inventory optimization. By collaborating with multiple suppliers, healthcare institutions can ensure a steady supply chain, negotiate better pricing, and mitigate risks of shortages due to supplier failures.

The implementation of advanced analytics and predictive will further optimize inventory levels, helping healthcare institutions balance supply and demand dynamically. Effective inventory management in hospital supply chains is crucial to ensuring the continuous availability of essential medicines while minimizing costs and waste. The study highlights the importance of demand forecasting, data-driven inventory control, and supply chain coordination to mitigate drug shortages. By implementing optimization techniques such as machine learning models, real-time tracking, and advanced replenishment strategies, hospitals can enhance efficiency and patient safety. Additionally, integrating Just-in-Time (JIT) inventory systems and supplier collaboration can further reduce risks associated with stockouts.

CHAPER 11
FUTURE ENHANCEMENT

Medical inventory optimization is crucial for ensuring the timely availability of essential medical supplies while minimizing waste and reducing costs. As technology advances and healthcare systems evolve, several future enhancements can further streamline inventory management. This document explores various innovative approaches that can enhance the efficiency, accuracy, and responsiveness of medical inventory systems.

1. AI-Driven Predictive Analytics

- ➢ Advanced machine learning models to predict demand fluctuations with higher accuracy.
- ➢ AI-powered insights to adjust stock levels dynamically based on trends, seasonality, and external factors like pandemics.

Enhancement:

- ➢ Integration with Electronic Health Records (EHRs) for real-time patient data analysis.
- ➢ AI-powered forecasting to prevent overstocking or understocking.

2. Blockchain for Secure & Transparent Supply Chain

- ➢ Blockchain-based tracking for end-to-end visibility of medical inventory.
- ➢ Reduction in counterfeit medicines and supply chain fraud.

Enhancement:

- ➢ Smart contracts for automated restocking and vendor agreements.
- ➢ Secure patient-linked prescription tracking to prevent misuse.

3.Robotic Process Automation (RPA) for Inventory Management

- Automated procurement, stock tracking, and reporting to reduce manual errors.
- AI-driven chatbots to handle supplier communications and reorders.

Enhancement:

- AI-powered voice assistants to monitor stock and place orders.
- Autonomous robots in warehouses for real-time inventory movement.

4.Augmented Reality (AR) for Warehouse & Inventory Optimization

- AR-assisted picking and restocking to improve accuracy and reduce errors.
- Smart glasses for warehouse staff to locate items quickly.

Enhancement:

- AI-powered vision systems for autonomous warehouse management.
- AR-based training modules for inventory staff.

5.3D Printing of Medical Supplies On-Demand

- On-site 3D printing of essential medical equipment (e.g., prosthetics, surgical tools).
- Reducing reliance on suppliers for critical, low-volume medical items.

Enhancement:

- AI-powered optimization of 3D printing for cost and efficiency.
- Integration with robotic surgery and emergency medicine.

6.Sustainable & Green Inventory Practices

- AI-driven sustainability analytics to minimize medical waste.
- Eco-friendly disposal methods for expired medicines and biohazardous materials.

Enhancement:

- AI-powered waste prediction and recycling recommendations.
- Blockchain-enabled tracking of medical waste disposal compliance.

7. Cloud-Based AI Inventory Platforms

- Real-time data sharing between multiple healthcare facilities for inventory pooling.
- Cloud-based dashboards for predictive insights and cross-location optimization.

Enhancement:

- AI-powered emergency response planning.
- Cloud-integrated multi-hospital inventory collaboration.

CHAPTER 12
REFERENCE

1. Kaakeh, R.; Sweet, B.V.; Reilly, C.; Bush, C.; DeLoach, S.; Higgins, B.; Clark, A.M.; Stevenson, Impact of drug shortages on U.S. health systems. Am. J. Health Syst. Pharm. 2011, 68, 1811– 1819.
   [Google Scholar] [CrossRef]

2. Fox, E.R.; Burgunda, V. Sweet, and Valerie Jensen, Drug shortages: A complex health care crisis. Mayo Clin. Proc. 2014, 89, 361–373. [Google Scholar] [CrossRef] [Green Version]

3. Stevenson, W.J.; Hojati, M.; Cao, J. Operations Management, 6th ed.; Stevenson explains this well in chapter 12 page 455; Thomson Reuters: Toronto, ON, Canada, 2018. [Google Scholar]

4. Aptel, O., & Pourjalali, H. (2001). Improving activities and decreasing costs of logistics in hospitals. A comparison of U.S. and French hospitals. The International Journal of Accounting, 36, 65–90.(Open in a new window)Google Scholar

5. Beaulieu, M., & Landry, S. (2002). Comment gérer la logistique hospitalière? Deux pays, deux réalités. Gestion, 27, 91–98.

6. Belien, J., & Demeulemeester, E. (2007). Building cyclic master surgery schedules with leveled resulting bed occupancy. European Journal of Operational Research, 176, 1185–1204. (Open in a new window)Google Scholar

7. Ahmed W., Huma S. Impact of lean and agile strategies on supply chain risk management. Total Qual. Manage. Business Excell. 2021;32:33–56. [Google Scholar]

8. Amoako-Gyampah K., Boakye K.G., Adaku E., et al. Supplier relationship management and firm performance in developing economies: a moderated mediation analysis of flexibility capability and ownership structure. Int. J. Prod. Econ. 2019;208:160–170. [Google Scholar]

9. Andaneswari, A. K. and Q. Rohmadiena, The Disruption of Personal Protection Equipment Supply Chain: What Can We Learn from Global Value Chain in the Time of Covid-19 Outbreak? Global South Review. 2, 171-188.

10. Aptel O., Pourjalali H. Improving activities and decreasing costs of logistics in hospitals: a comparison of US and French hospitals. Int. J. Account. 2001;36:65–90. [Google Scholar]

11. Kaakeh, R.; Sweet, B.V.; Reilly, C.; Bush, C.; DeLoach, S.; Higgins, B.; Clark, A.M.; Stevenson, Impact of drug shortages on U.S. health systems. Am. J. Health Syst. Pharm. 2011, 68, 1811– 1819. [Google Scholar] [CrossRef]

12. Fox, E.R.; Burgunda, V. Sweet, and Valerie Jensen, Drug shortages: A complex health care crisis. Mayo Clin. Proc. 2014, 89, 361–373. [Google Scholar] [CrossRef] [Green Version]

13. Stevenson, W.J.; Hojati, M.; Cao, J. Operations Management, 6th ed.; Stevenson explains this well in chapter 12 page 455; Thomson Reuters: Toronto, ON, Canada, 2018. [Google Scholar]

14. Wu I.C., Lin Y.C., Yien H.W., Shih F.Y. Constructing constraint-based simulation system for creating emergency evacuation plans: A case of an outpatient chemotherapy area at a cancer medical center. Healthcare. 2020;8:137. doi: 10.3390/healthcare8020137. [DOI] [PMC free article] [PubMed] [Google Scholar]

15. Liu Z.Y. The anti-hepatitis drug use effect and inventory management optimization from the perspective of hospital drug supply chain. Pak. J. Pharm. Sci. 2017;30:1917–1922. [PubMed] [Google Scholar]

16. Alves R.J.V., Etges A., Neto G.B., Polanczyk C.A. Activity-based costing and time-driven activitybased costing for assessing the costs of cancer prevention, diagnosis, and treatment: A systematic review of the literature. Value Health Reg. Issues. 2018;17:142–147. doi: 10.1016/j.vhri.2018.06.001. [DOI] [PubMed] [Google Scholar]

17. Abbas S.A., Aslam A., Rehman A.U., Abbasi W.A., Arif S., Kazmi S.Z.H. K-means and kmedoids: Cluster analysis on birth data collected in city Muzaffarabad, Kashmir. IEEE Access. 2020;8:151847–151855. doi: 10.1109/ACCESS.2020.3014021. [DOI] [Google Scholar]