**Department of Electronics and Communication Engineering**  
**ADITYA COLLEGE OF ENGINEERING AND TECHNOLOGY**  
 Surampalem, Andhra Pradesh

**MEDICAL INVENTORY MANAGEMENT**



**Submitted by**

[M. SRIVIDYA] - [ 23P31A04H0]

[VATTEM SUDHEER] - [23P31A04D3]

[VATTEM LAKSHMAN] - [23P31A04C8]

[VUDAVALLI DWARAKANADH] - [23P31A0466]

**Submitted to**  
[Mr. Anji Babu]  
*Project Guide, Dept. of ECE*

**June 2025**

**ACKNOWLEDGEMENT**

We take immense pleasure in expressing our heartfelt gratitude to all those who contributed directly or indirectly to the successful completion of our internship project titled **“Medical Inventory Management.”**

First and foremost, we are sincerely thankful to our esteemed project guide, **[ANJAI BABU]**, for their continuous support, expert guidance, and valuable suggestions throughout the development of this project. Their insights and encouragement were crucial in transforming our ideas into a functional solution.

We extend our sincere thanks to the **Head of the Department, Department of Electronics and Communication Engineering**, and all faculty members of **Aditya College of Engineering and Technology**, for providing us with the platform, encouragement, and resources necessary to undertake this project.

Our heartfelt gratitude also goes to **Smartbridge** for offering us the opportunity to work on this real-time, industry-relevant platform. Their mentorship and feedback played a pivotal role in shaping the quality and direction of our work.

We are especially thankful to our team members, whose collaboration, commitment, and enthusiasm helped us overcome challenges and meet our project goals:

* (M. Srividya ),
* ( Vattem Sudheer) ,
* (Vattem Lakshman) ,
* ( Vundavalli Dwarakanadh).

Lastly, we would like to thank our **parents, friends, and well-wishers** for their continuous encouragement, motivation, and moral support throughout this journey.

This project has helped us not only strengthen our technical abilities but also improve our teamwork, communication, and problem-solving skills. It has been a valuable learning experience, and we are proud of what we have achieved together.

**TABLE OF CONTENTS**

| **S. No.** | **Content** | **Page No.** |
| --- | --- | --- |
| 1. | **COVER PAGE** | 1 |
|  | **ACKNOWLEDGEMENT** | 2 |
| 3. | **TABLE OF CONTENTS** | 3 |

4. **INTRODUTION** 4–5

4.1 Project Overview

5. **IDEACTION PHASE** 6-7

5.1 Problem Statement

5.2 Empathy Map Canvas

5.3 Brainstorming Solution

6. **REQUIREMENT ANALYSIS** 8-11

6.1 Customer Journey Map

6.2 Solution Requirement

6.3 Data Flow Diagram

7. **PROJECT DESIGN 1**2-14

7.1 Problem Solution Fit

7.2 Proposed Solution

7.3 Solution Architecture

8. **PROJECT PLANNING & SCHHEDULING** 15-16

8.1 Project Planning

9.  **FUNCTIONAL AND PERFORMING TESTING** 17-18

9.1 Performance Testing

10. RESULT 19-23

10. Output Screenshots

11. **ADVANTAGES & DISADVANTAGES** 24-25

12. **CONCLUSION** 26

13. **FUTURE SCOPE** 27

14.  **APPENDIX** 28-30

Source Code

GitHub & Project Demo Link

**INTRODUCTION**

The **Medical Inventory Management System** is a Salesforce-based application tailored to manage and streamline various operational aspects of medical inventory in hospitals or clinics with a digital solution to manage their medical inventory efficiently. Traditional systems often rely on manual data entry and fragmented records, leading to delays, stockouts, expired medicines, and human error.

Hospital inventory managers, pharmacists, and administrative staff often face daily challenges such as lost supplier contact details, forgotten expiry dates of medicines, incorrect stock levels, and lack of visibility into supply orders. These problems can directly impact patient safety, compliance, and the smooth functioning of medical services.

Users need a centralized, easy-to-use system that manages end-to-end inventory tasks. It should keep track of suppliers, ensure the availability of products, highlight urgent restocking needs, and support data-driven decisions with comprehensive reports. The key pain points are: manual tracking, no expiry alerts, unorganized purchase data, and lack of real-time insights.

To address these challenges, we propose a **Salesforce-based Medical Inventory Management System** that includes Supplier management with contact and business details Creation and management of purchase orders Real-time tracking of medical products and transactions Automatic alerts for low stock or approaching expiry dates Reports and dashboards to monitor operations and trends

This Salesforce-based solution simplifies the traditionally manual and error-prone process of managing inventory by:

* **Centralizing supplier, product, and purchase order information**
* **Automating inventory tracking and updates**
* **Monitoring expiry dates to prevent usage of outdated products**
* **Generating real-time reports and dashboards** for better decision-making

By digitizing and automating these operations, the system ensures timely stock replenishment, minimizes wastage, and supports compliance with safety and quality standards. Its role-based access and process automation make it scalable, secure, and adaptable to the needs of any healthcare facility.

**4.1 Project Overview**

The **Medical Inventory Management System** is a custom-built Salesforce application that revolutionizes how healthcare facilities handle medical supply chains. By leveraging Salesforce’s robust platform capabilities, this solution enables end-to-end digitization and automation of inventory processes, minimizing manual effort and human errors. At its core, the system features an intuitive **Supplier Management** module that stores essential supplier details, such as contact numbers, email addresses, and company profiles, streamlining communication and procurement workflows.

It also offers **Product Tracking**, where every medical item is logged with specific attributes like description, quantity, batch number, and expiry date, ensuring accurate stock visibility and compliance with safety protocols. The **Purchase Order** functionality enables the creation, monitoring, and management of transactions with suppliers, offering traceability from order to delivery. With real-time **Inventory Monitoring**, users are alerted when stock levels fall below critical thresholds or when products are nearing their expiration dates, allowing timely restocking and safe usage of medical items.

To support data-driven decision-making, the system provides powerful **Reporting and Dashboard** features, offering visual insights into supplier performance, order frequency, inventory trends, and expiration forecasts. These analytics help optimize inventory turnover, reduce wastage, and ensure uninterrupted patient care. Overall, the application empowers hospital administrators, pharmacists, and inventory managers with a centralized, cloud-based platform that enhances efficiency, accuracy, and accountability across the medical inventory lifecycle.

**IDEATION PHASE**

The **Ideation Phase** is the foundation of any successful project. It focuses on identifying real-world problems, brainstorming potential solutions, and narrowing down the idea that best aligns with user needs and technical feasibility. For the **Medical Inventory Management System**, this phase was critical in ensuring the solution addresses key challenges faced by hospitals and clinics in managing their inventory.

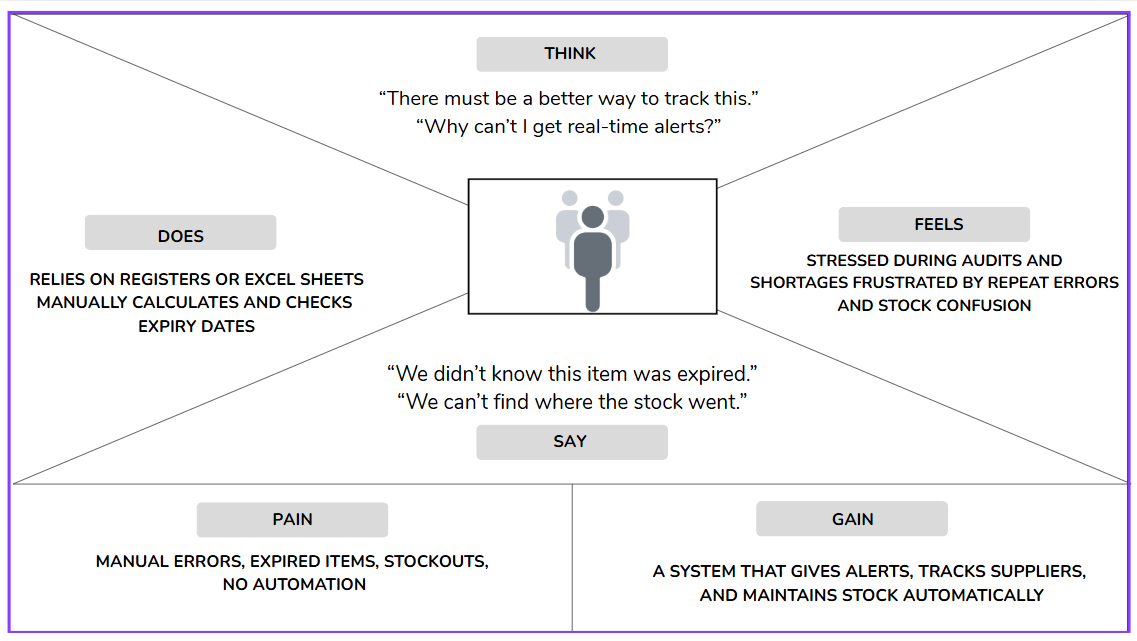
**5.1 Problem Statement**

In many hospitals and clinics, inventory management is still handled using manual methods such as registers or basic spreadsheets. These outdated processes lead to significant challenges:

* **Expired medical supplies** being administered to patients
* **Stockouts** of essential medicines or equipment
* **Inability to track supplier performance**
* **No real-time visibility** into current inventory levels
* **Lack of automated alerts** for restocking or expiry

**5.2 Empathy Map Canvas**

To better understand the end-users—mainly inventory managers, medical staff, and administrators—we created an empathy map. This tool helps in viewing the problem from the user’s perspective, which is essential for designing user-friendly solutions.



**5.3 Brainstorming Solution**

The brainstorming session included research, user interviews, and team discussions. Several ideas were evaluated before settling on a Salesforce-based solution. Key decisions made during this phase included:

* Use **Salesforce Lightning** to create a responsive and secure cloud application.
* Create **custom objects** for Product, Supplier, Purchase Order, and Transactions.
* Use **Flows** and **Apex Triggers** for automation of stock updates and alerting mechanisms.
* Design intuitive **Lightning Pages** and **Dashboards** for different user roles.
* Implement **Profiles and Permission Sets** for secure access management.

We also compared the solution with other platforms (like Excel-based tools and low-code apps) but found Salesforce best suited for scalability, automation, security, and data-driven decision-making.

**5.4 Goal Setting**

Clear goals were defined to guide the project development:

* Create a scalable and secure inventory management app on Salesforce
* Automate critical processes (like expiry alerts and stock updates)
* Provide clear data visualization for inventory status and supplier performance
* Ensure ease of use for different roles (Admin, Manager, Staff)

**5.5 Project Planning**

Finally, we divided the project into structured **milestones**, each covering an essential part of the system—starting from account creation and object setup, to advanced automation and dashboards.

**Outcome of the Ideation Phase**

The ideation phase successfully established the scope, identified the user needs, selected the right platform (Salesforce), and created a roadmap to build a powerful, real-time, and user-friendly **Medical Inventory Management System**. This phase ensured that the project stayed user-focused, technically feasible, and aligned with real-world healthcare inventory challenges.

**REQUIREMENT ANALYSIS**

The **Requirement Analysis** phase plays a critical role in translating the idea into a structured and actionable development plan. It involves gathering, analyzing, and documenting all the functional and non-functional requirements necessary to build a robust, scalable, and user-friendly **Medical Inventory Management System** using Salesforce.

**6.1. Functional Requirements**

These are the specific functionalities that the system must perform to meet the needs of its users.

**6.1.1 Supplier Management**

* Create, view, update, and delete supplier records.
* Store supplier contact information (Name, Email, Phone, Address).
* Track supplier performance through associated purchase orders.

**6.1.2 Product Management**

* Add and manage product details (Name, Description, Category).
* Record current and minimum stock levels.
* Track expiry dates and unit prices of each product.

**6.1.3 Purchase Order Management**

* Generate purchase orders linked to suppliers and products.
* Record order date, expected delivery date, and total amount.
* Monitor the status of each order (Pending, Received, Cancelled).

**6.1.4 Inventory Transactions**

* Record stock inflow (purchase) and outflow (usage).
* Automatically update stock levels after each transaction.
* Link transactions to products and relevant purchase orders.

**6.1.5 Alerts & Notifications**

* Alert users when stock falls below minimum threshold.
* Notify managers when a product is nearing expiry or expired.

**6.1.6 User Management**

* Create and assign roles (Admin, Inventory Manager, Medical Staff).
* Implement role-based access control using Profiles and Permission Sets.

**6.1.7 Automation**

* Flows to automate stock updates and expiry notifications.
* Apex triggers for real-time stock adjustment based on transactions.

**6.1.8 Reports and Dashboards**

* Generate reports on stock status, expired products, purchase trends.
* Create dashboards for at-a-glance monitoring of inventory health and supplier efficiency.

**6.2. Non-Functional Requirements**

These describe system qualities and constraints that are not directly related to specific behaviors.

**6.2.1 Usability**

* The system should have a simple and intuitive user interface using Lightning Pages.
* Each user role should have customized views relevant to their responsibilities.

**6.2.2 Reliability**

* Data must be accurately maintained and consistently updated in real-time.
* Triggers and flows must handle exceptions gracefully (e.g., negative stock levels).

**6.2.3 Performance**

* Reports and dashboards should load within a few seconds for real-time monitoring.
* Inventory updates via triggers and flows should be seamless and fast.

**6.2.4 Scalability**

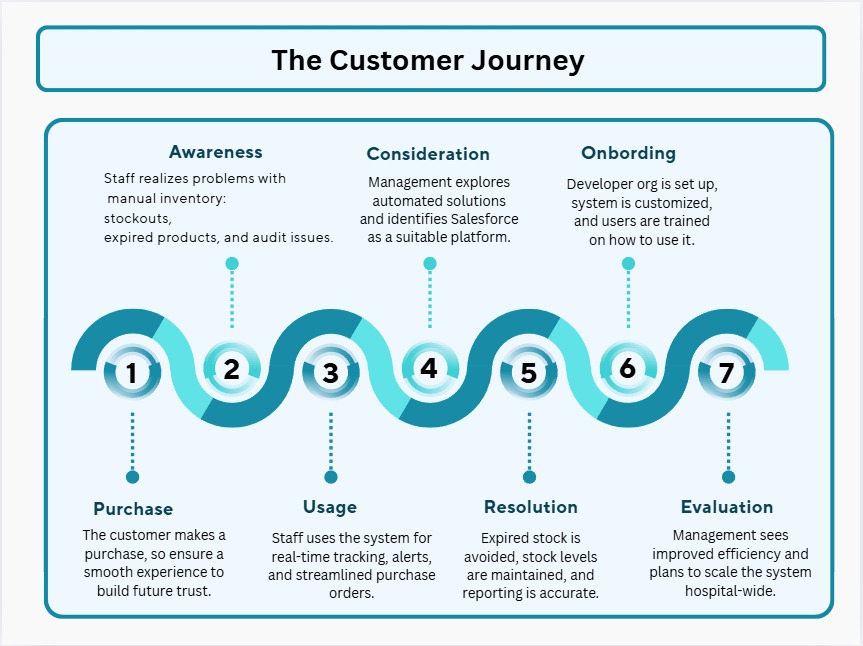
* The system should be able to accommodate increasing numbers of products, suppliers, and users without performance degradation.

**6.2.5 Security**

* Sensitive data (e.g., supplier pricing) should be protected via field-level security.
* Profiles and permission sets must ensure appropriate access control.

**6.2.6 Accessibility**

* Should be accessible from any device via Salesforce m



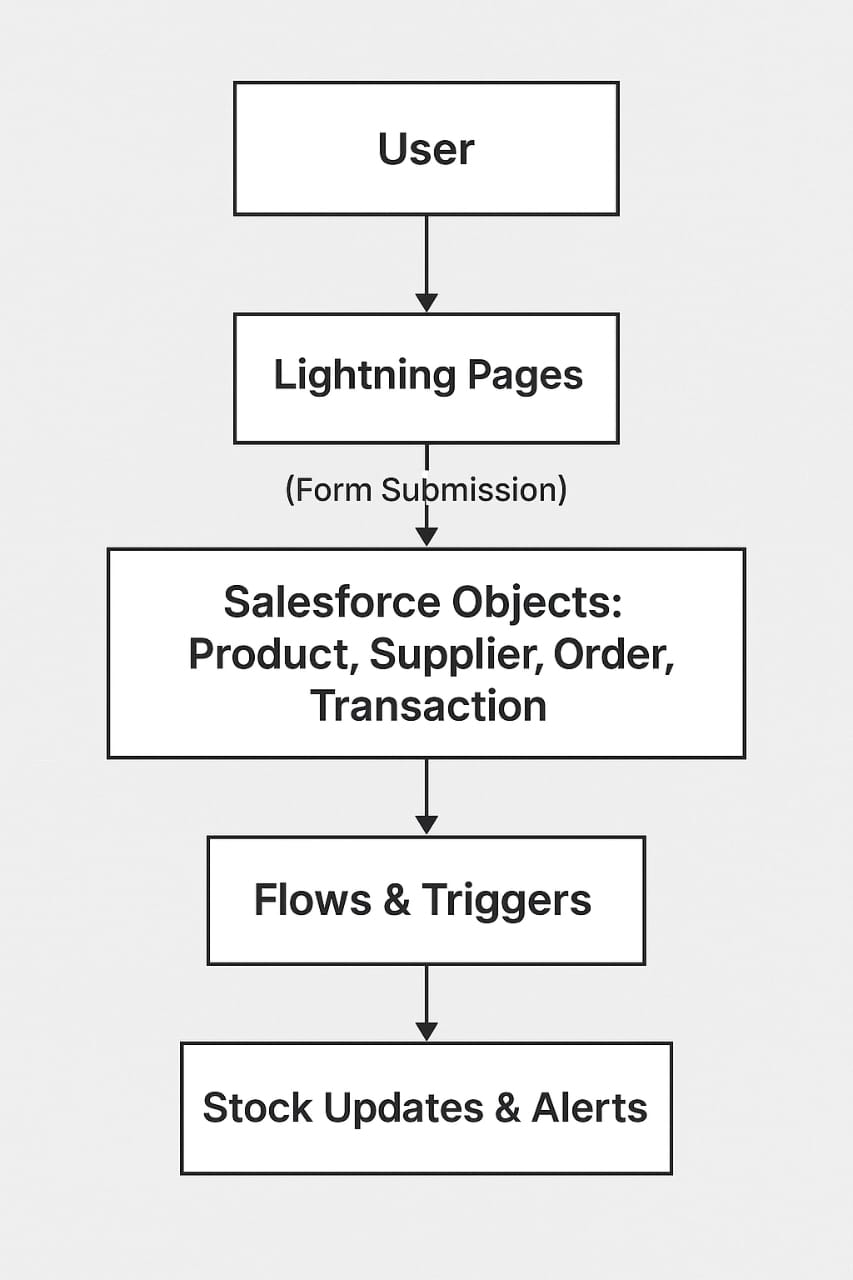
**6.3 Data Flow Diagram (Level 0)**

Users interact via UI (tabs, forms).

Data flows to custom objects (Product, Supplier, etc.).

Flows & triggers automate stock updates and alerts.

Reports & dashboards visualize current state.



**6.4 Technology Stack**

The project is built entirely within the Salesforce ecosystem, leveraging its powerful cloud-based tools for database management, UI creation, automation, and analytics. Below is a breakdown of each component in the stack and its role in the system:

**6.4.1. Custom Objects & Fields**

* **Type**: Salesforce Data Layer
* **Purpose**: Core data structure to represent inventory entities like:
  + Product\_\_c: Product name, unit price, current stock, expiry date
  + Supplier\_\_c: Contact and vendor details
  + Purchase\_Order\_\_c: Order details linked to suppliers and products
  + Inventory\_Transaction\_\_c: Stock in/out records linked to products
* **Why Used**: Enables flexible, scalable, and relational data modeling.

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** |  | **Technology** | **Purpose** |
| **Platform** |  | Salesforce Lightning | CRM, cloud hosting, and application base |
| **Automation** |  | Salesforce Flows, Apex Triggers | Workflow logic, automation |
| **UI** |  | Lightning Pages, Tabs | User Interface |
| **Data Modeling** |  | Custom Objects & Fields | Storing structured information |
| **Security** |  | Roles, Profiles, Permission Sets | Access control |
| **Reporting** |  | Salesforce Reports & Dashboards | Analytics and performance monitoring |
| **Testing** |  | Salesforce Sandbox | Pre-deployment testing environment |

**PROJECT DESIGN**

Hospitals and clinics rely on a steady and safe supply of medicines and equipment. However, most healthcare facilities still depend on **manual processes** or disconnected digital tools (like spreadsheets), which lead to several critical challenges:

1. **Manual Tracking & Human Errors**  
   Inventory is recorded manually, which increases the risk of mistakes such as incorrect entries, duplication, or oversight. These errors can result in overstocking or understocking.
2. **Lack of Real-Time Visibility**  
   Staff often have no live view of current inventory levels. This makes it difficult to make quick decisions during emergencies or procurement planning.
3. **Missed Alerts for Expiry or Stockouts**  
   Without automated notifications, medical staff may unknowingly use **expired products** or discover **stockouts** when it’s too late, putting patient safety at risk.
4. **Poor Supplier Accountability**  
   Manual processes don’t provide insights into supplier performance (e.g., delivery delays, order accuracy), which affects planning and budgeting.
5. **Inefficient Reporting and Audits**  
   During inspections or audits, staff struggle to produce historical data, usage logs, or compliance documents due to fragmented and inconsistent records.

**7.1 The Need**

To address these problems, there is a **critical need** for a reliable and automated inventory management system built specifically for medical use cases. The ideal solution should:

1. **Enable Real-Time Inventory Monitoring**  
   Automatically track product availability, stock levels, and movements without manual input.
2. **Send Timely Alerts & Notifications**  
   Notify users when stock levels fall below minimum thresholds or when products are nearing expiry.
3. **Improve Transparency and Reduce Wastage**  
   Ensure accountability for every product purchased, stored, or used—minimizing both overstock and expired items.
4. **Maintain Accurate Records**  
   Keep well-structured, searchable records of **suppliers, purchase orders, inventory transactions**, and **product life cycles** for audits and reports.

**Fit with the Salesforce Platform**

Salesforce provides a robust, secure, and highly customizable platform that supports:

* Real-time data updates through automation
* Centralized record management
* Custom alerts and workflows
* Role-based access controls
* Built-in reporting and dashboard capabilities

This makes Salesforce a perfect fit for solving the medical inventory problems described above.

**7.2 Proposed Solution**

To address the problems identified, we propose a **Salesforce-based Medical Inventory Management System** that automates and centralizes inventory operations in healthcare facilities.

**Key Features**

* **Custom Objects**:
  + Product\_\_c, Supplier\_\_c, Purchase\_Order\_\_c, and Inventory\_Transaction\_\_c for structured data management.
* **Real-Time Automation**:
  + **Flows** to auto-update stock levels and trigger expiry alerts.
  + **Apex Triggers** for accurate transaction-based stock updates.
* **Validation Rules**:
  + Enforce business logic (e.g., expiry date must be in the future, stock can’t be negative).
* **Access Control**:
  + **Roles** and **Profiles** define user privileges.
  + **Permission Sets** grant additional access as needed.
* **Dashboards & Reports**:
  + Real-time visualizations for inventory status, expired products, and supplier efficiency.

**Benefits**

* Eliminates manual entry errors
* Prevents expired stock usage
* Provides real-time stock visibility
* Ensures accountability and audit readiness
* Helps meet healthcare safety and compliance standards

**7.3 Solution Architecture**

The solution is designed using a **multi-layered architecture** that aligns with Salesforce’s best practices:

**1. Presentation Layer (UI)**

* **Lightning Tabs & Pages** for navigation
* **Compact Layouts** for quick data access
* Responsive design usable across devices

**2. Business Logic Layer**

* **Flows** for inventory automation
* **Apex Triggers** for transactional logic
* **Validation Rules** for ensuring clean data

**3. Data Layer**

* **Custom Objects** to store and relate data:
  + Products → Transactions
  + Suppliers → Purchase Orders

**4. Security Layer**

* **Profiles**, **Roles**, and **Permission Sets** for secure, role-based data access

**5. Reporting Layer**

* **Reports**: Expired Products, Stock Levels, Purchase Order Summary
* **Dashboards**: Visual charts for stock trends and supplier performance

A diagram of a business process

AI-generated content may be incorrect.

**PROJECT PLANNING & SCHEDULING**

Effective planning and scheduling are critical for the successful development and deployment of the Medical Inventory Management System. The goal is to ensure that each phase of the project is executed within a defined timeline, with clearly assigned tasks, dependencies, and outcomes.

**8.1 Project Planning Objectives**

* Define milestones based on functional development stages.
* Allocate time for configuration, automation, testing, and documentation.
* Ensure smooth transitions between planning, design, development, and testing phases.
* Assign specific responsibilities to team members (if applicable).

**8.2 Project Development Phases & Milestones**

| **Milestone No.** | **Milestone Name** | **Activities Covered** | **Timeline** |
| --- | --- | --- | --- |
| M1 | Developer Org Setup | Create Salesforce Developer Org, assign admin access | Day 1 |
| M2 | Object Creation | Create custom objects: Product, Supplier, Purchase Order, Inventory Transaction | Days 2–3 |
| M3 | Tab and Lightning App Setup | Create tabs for objects and configure Lightning App | Day 4 |
| M4 | Field Configuration | Add custom fields to objects (e.g., expiry date, stock level, supplier name) | Days 5–6 |
| M5 | Page & Compact Layouts | Update page layouts and compact layouts for all objects | Day 7 |
| M6 | Validation Rules Implementation | Create and test validation rules (e.g., expiry date > today, positive quantity) | Day 8 |
| M7 | Role & Profile Setup | Define roles (Admin, Manager, Staff) and assign profiles | Day 9 |
| M8 | User & Permission Sets | Create test users and apply relevant permission sets | Day 10 |
| M9 | Flows Creation | Build flows for stock update and expiry alerts | Days 11–12 |
| M10 | Apex Trigger Development | Create Apex triggers for real-time stock calculations | Days 13–14 |
| M11 | Reports and Dashboards | Create inventory reports and visual dashboards for management | Days 15–16 |
| M12 | Functional Testing | Validate all object relationships, flows, and reports | Days 17–18 |
| M13 | Documentation & Submission | Finalize documentation, screenshots, and project files | Days 19–20 |

**8.3 Tools Used for Planning and Execution**

| **Tool** | **Purpose** |
| --- | --- |
| Salesforce Developer Org | Platform for development and testing |
| Lucidchart / Draw.io | ER diagrams, architecture diagrams |
| Google Docs / Word | Documentation and report creation |
| Google Sheets / Excel | Planning timeline, task tracking |
| PowerPoint | Presentation and final demo slides |

**8.4 Key Success Indicators**

* All milestones completed within the allocated timeline.
* System is fully functional with real-time automation and accurate reporting.
* Users are able to interact with the system based on roles and permissions.
* Reports and dashboards accurately reflect inventory data and metrics.

**FUNCTIONAL AND PERFORMANCE TESTING**

**9.1 Functional Testing**

Functional testing ensures that every component of the system works according to the specified requirements. In this project, it validates how well the **Medical Inventory Management System** performs its intended operations across different objects, workflows, and users.

**Test Objectives**

* Confirm all custom objects (Product, Supplier, Purchase Order, Inventory Transaction) function correctly.
* Validate field-level behaviors, including validation rules.
* Ensure automation (Flows, Triggers) executes as expected.
* Verify access control via roles, profiles, and permission sets.
* Check that reports and dashboards display correct data.
* Ensure UI components (Lightning Pages, Tabs) are user-friendly and properly linked.

**Functional Test Cases**

| **Test Case** | **Description** | **Expected Result** | **Status** |
| --- | --- | --- | --- |
| TC01 | Create a new Product record | Record saved with default stock level | Pass |
| TC02 | Create a Purchase Order with a future delivery date | Record created and linked to supplier | Pass |
| TC03 | Attempt to enter expired product | Validation rule prevents save | Pass |
| TC04 | Inventory Transaction reduces stock count | Product stock updated automatically | Pass |
| TC05 | Flow triggers alert when stock < threshold | Notification/Email sent | Pass |
| TC06 | Access Product object with Manager profile | Allowed based on profile settings | Pass |
| TC07 | Staff user cannot delete Purchase Orders | Permission restricted | Pass |
| TC08 | Dashboard displays current stock and expiry data | Accurate and real-time | Pass |
| TC09 | Run report for all expired products | List shows correct filtered records | Pass |

**Access & Security Testing**

| **Role** | **Tested For** | **Result** |
| --- | --- | --- |
| Admin | Full create/read/edit/delete access | Pass |
| Manager | Can manage orders and inventory, but limited user access | Pass |
| Staff | Can view stock and place purchase orders only | Pass |

**UI Functionality Testing**

* Tabs load correctly and link to relevant records
* Record pages display related lists properly
* Quick actions (e.g., Add Transaction) work on Lightning App
* Field visibility changes based on profiles

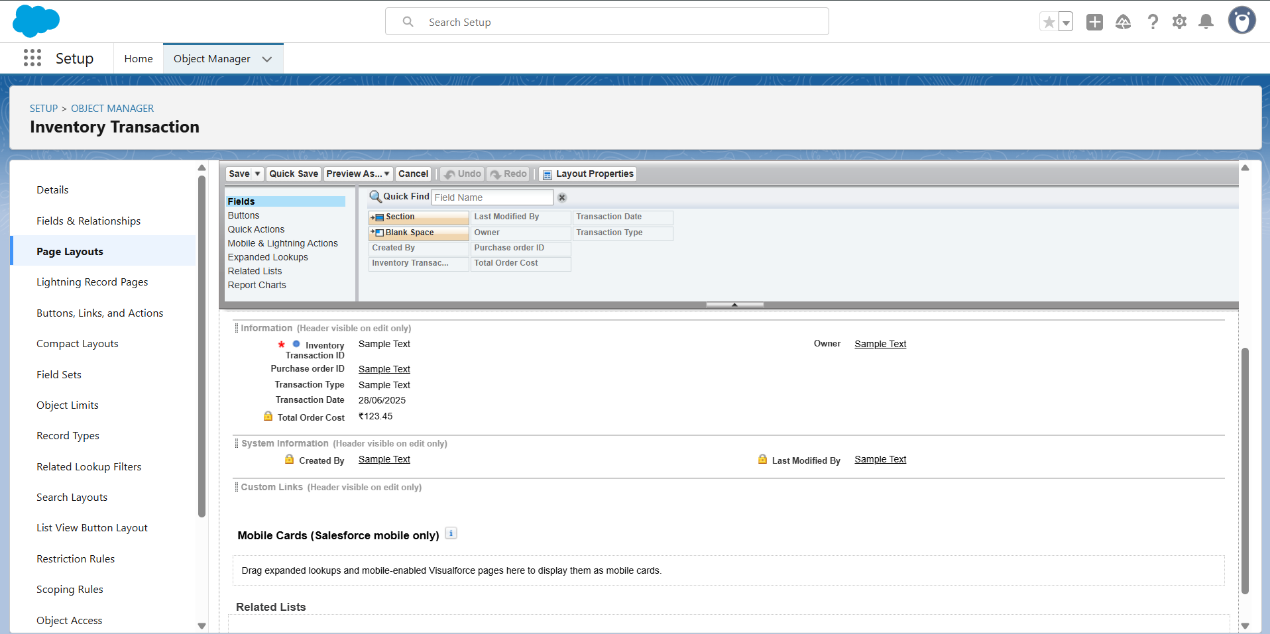
**Conclusion**

Functional testing confirms that the system performs as expected, with:

* Accurate stock updates
* Role-based access working correctly
* Automation functioning as designed
* Reports and dashboards reflecting real-time, reliable data

The system is now **functionally ready for user training and deployment.**

**RESULTS**



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

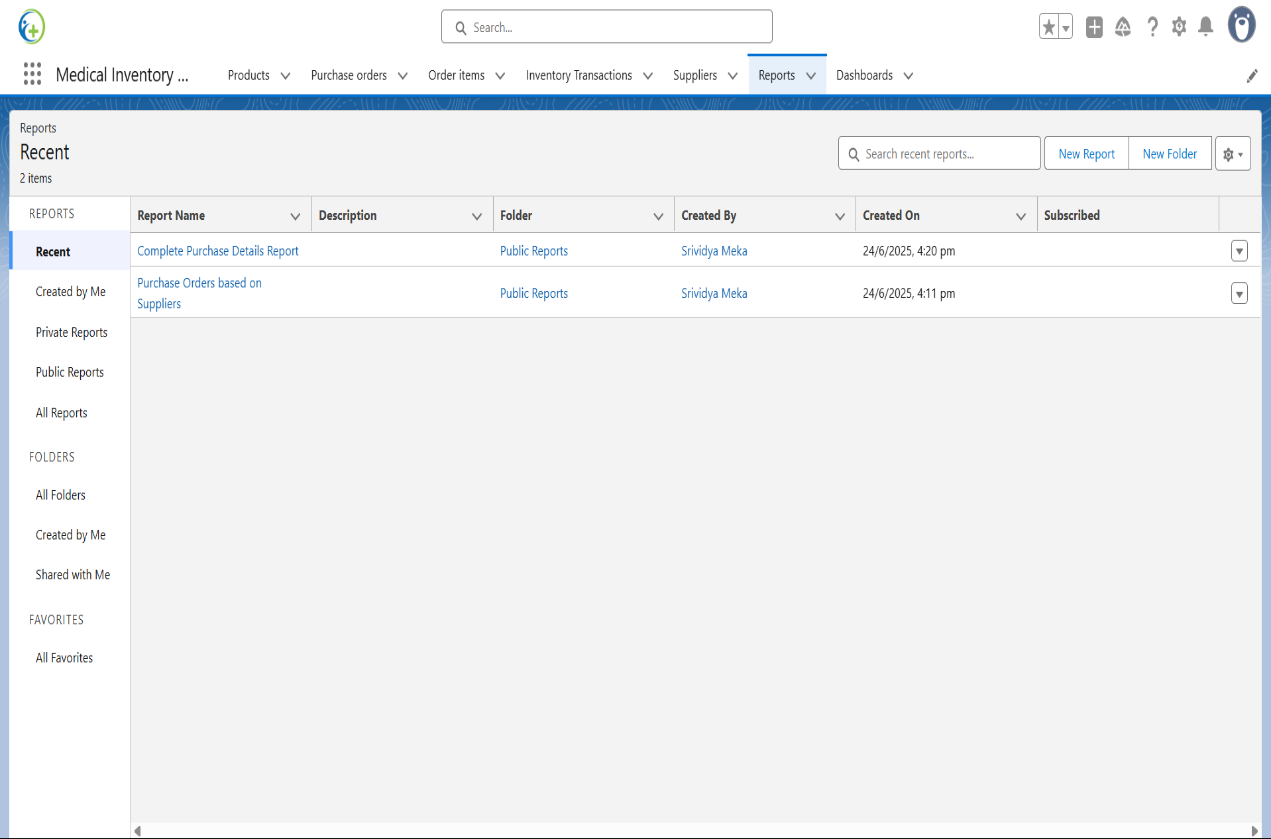
AI-generated content may be incorrect.

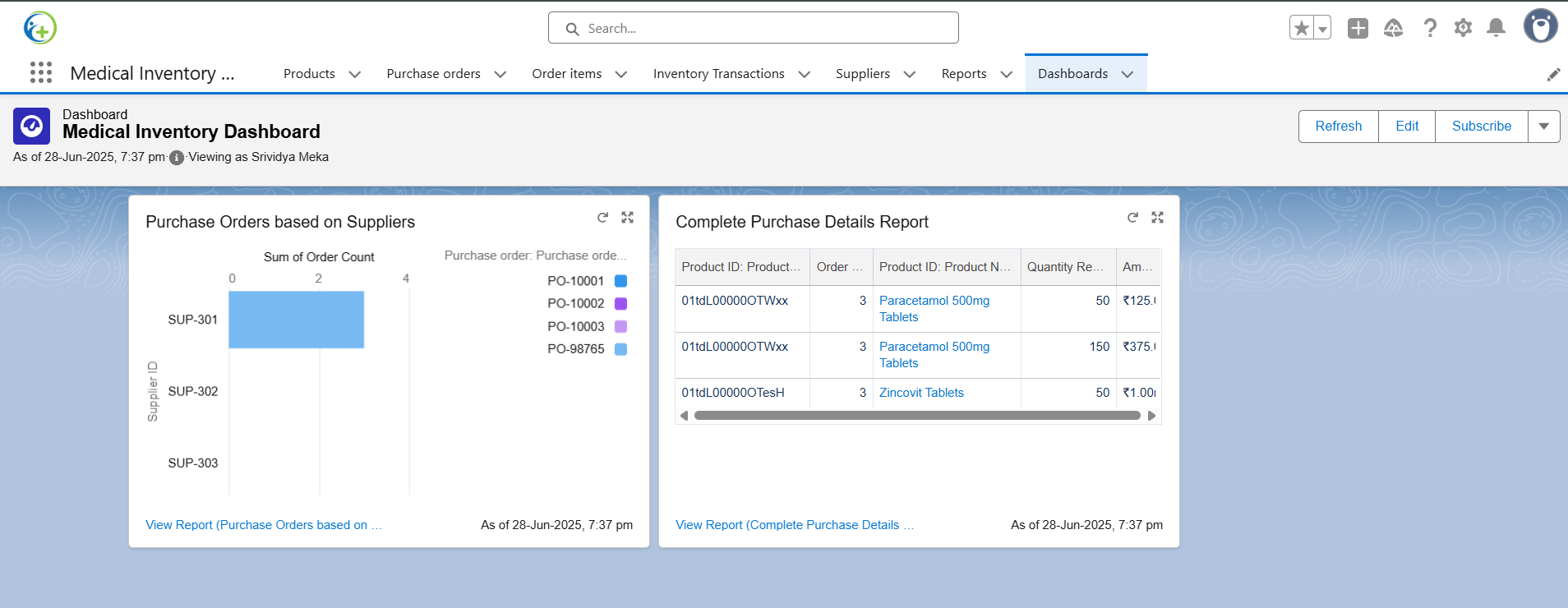
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.





**ADVANTAGES & DISADVANTAGES**

**Advantages**

1. **Real-Time Inventory Management**  
   Automatically tracks stock levels, product movements, and expiry dates without manual intervention.
2. **Improved Accuracy & Automation**  
   Reduces human errors using validation rules, flows, and Apex triggers for consistent and accurate data.
3. **Timely Alerts & Notifications**  
   Sends proactive alerts for low stock and upcoming expiries, ensuring product availability and safety.
4. **Centralized Data Management**  
   All supplier, product, and transaction data are stored in a single Salesforce platform—easy to access, update, and audit.
5. **Role-Based Access Control**  
   Customizable permissions using roles, profiles, and permission sets to ensure data security.
6. **Easy Reporting and Dashboards**  
   Built-in tools help generate detailed reports and interactive dashboards for real-time decision-making.
7. **Scalability and Cloud Access**  
   Salesforce allows the system to scale with growing needs and access data from anywhere via cloud.
8. **Compliance-Ready System**  
   Ensures that expired items are not used and every transaction is recorded for audit purposes.

**Disadvantages**

1. **Initial Setup Complexity**  
   Requires technical knowledge to configure objects, automation, and access controls correctly.
2. **Learning Curve**  
   Users unfamiliar with Salesforce may need training to navigate the Lightning interface and features.
3. **Internet Dependency**  
   Since the system is cloud-based, uninterrupted internet connectivity is essential for real-time usage.
4. **Cost of Licensing**  
   Salesforce licenses may be costly for small clinics or organizations with limited budgets.
5. **Customization Time**  
   Tailoring the system to meet very specific workflows (e.g., barcode scanning or integration with hospital ERP) may take extra development time.

**Summary**

| **Aspect** | **Pros** | **Cons** |
| --- | --- | --- |
| **Automation** | Reduces errors, improves speed | Initial setup requires time and planning |
| **Accessibility** | Cloud-based and available anytime, anywhere | Needs stable internet connection |
| **Usability** | Intuitive UI with dashboards | Some users may need onboarding |
| **Scalability** | Easily expandable for larger facilities | May require additional customization over time |
| **Security** | Role-based access ensures data protection | Misconfigured profiles could limit or expose data |

**CONCLUSION**

The **Medical Inventory Management System**, built on the Salesforce platform, provides a robust and scalable solution for managing medical supplies in healthcare facilities. By automating key processes such as stock tracking, supplier management, purchase ordering, and expiry monitoring, the system reduces manual errors and enhances operational efficiency.

This solution addresses critical problems in the healthcare inventory domain by offering:

* Real-time visibility into stock levels
* Automated alerts for low stock and expiring products
* Accurate and centralized record-keeping
* Custom dashboards and reports for data-driven decisions
* Role-based access to ensure security and compliance

Through a combination of custom objects, automation tools (flows, triggers), and user-focused design, this project demonstrates how technology can improve healthcare outcomes by ensuring the right medical resources are available at the right time.

In conclusion, this Salesforce-based system empowers healthcare providers to manage their inventory with greater confidence, safety, and control—ultimately supporting better patient care and resource utilization.

**FUTURE SCOPE**

While the current version of the Medical Inventory Management System provides a solid foundation for managing inventory within healthcare facilities, several enhancements can be explored to make the system even more powerful, intelligent, and user-friendly in the future:

**1. Barcode and QR Code Integration**

* Enable barcode/QR scanning for faster and error-free data entry of products, batch numbers, and transactions.
* Improves check-in/check-out processes in hospital storerooms.

**2. AI-Based Predictive Restocking**

* Implement AI/ML algorithms to predict stock consumption trends.
* Auto-generate purchase orders based on historical usage and seasonal demand patterns.

**3. Mobile App Interface**

* Create a mobile-friendly interface for on-the-go inventory updates, alerts, and approval workflows.
* Enables nurses, pharmacists, and procurement teams to work from handheld devices.

**4. Vendor API Integration**

* Direct integration with supplier APIs to automate procurement processes.
* Track delivery timelines, verify order details, and streamline reordering.

**5. IoT Device Support**

* Connect smart storage units or sensors to track temperature-sensitive medicines or detect expired items automatically.

**6. Multi-Location Inventory Management**

* Enable centralized control over inventories spread across multiple branches or hospital departments.

**7. Audit & Compliance Automation**

* Generate automated compliance reports for health authorities.
* Maintain logs of every inventory-related action for regulatory audits.

By implementing these future enhancements, the Medical Inventory Management System can evolve into a **comprehensive digital health supply chain platform**, supporting smarter decisions, better resource allocation, and improved patient care outcomes.

**APPENDIX**

**Source code :** trigger CalculateTotalAmountTrigger on Order\_Item\_\_c (after insert, after update, after delete, after undelete) {

// Call the handler class to handle the logic

CalculateTotalAmountHandler.calculateTotal(Trigger.new, Trigger.old, Trigger.isInsert, Trigger.isUpdate, Trigger.isDelete, Trigger.isUndelete);

}

public class CalculateTotalAmountHandler {

// Method to calculate the total amount for Purchase Orders based on related Order Items

public static void calculateTotal(List<Order\_Item\_c> newItems, List<Order\_Item\_c> oldItems, Boolean isInsert, Boolean isUpdate, Boolean isDelete, Boolean isUndelete) {

// Collect Purchase Order IDs affected by changes in Order\_Item\_\_c records

Set<Id> parentIds = new Set<Id>();

// For insert, update, and undelete scenarios

if (isInsert || isUpdate || isUndelete) {

for (Order\_Item\_\_c ordItem : newItems) {

parentIds.add(ordItem.Purchase\_Order\_Id\_\_c);

}

}

// For update and delete scenarios

if (isUpdate || isDelete) {

for (Order\_Item\_\_c ordItem : oldItems) {

parentIds.add(ordItem.Purchase\_Order\_Id\_\_c);

}

}

// Calculate the total amounts for affected Purchase Orders

Map<Id, Decimal> purchaseToUpdateMap = new Map<Id, Decimal>();

if (!parentIds.isEmpty()) {

// Perform an aggregate query to sum the Amount\_\_c for each Purchase Order

List<AggregateResult> aggrList = [

SELECT Purchase\_Order\_Id\_c, SUM(Amount\_c) totalAmount

FROM Order\_Item\_\_c

WHERE Purchase\_Order\_Id\_\_c IN :parentIds

GROUP BY Purchase\_Order\_Id\_\_c

];

// Map the result to Purchase Order IDs

for (AggregateResult aggr : aggrList) {

Id purchaseOrderId = (Id)aggr.get('Purchase\_Order\_Id\_\_c');

Decimal totalAmount = (Decimal)aggr.get('totalAmount');

purchaseToUpdateMap.put(purchaseOrderId, totalAmount);

}

// Prepare Purchase Order records for update

List<Purchase\_Order\_c> purchaseToUpdate = new List<Purchase\_Order\_c>();

for (Id purchaseOrderId : purchaseToUpdateMap.keySet()) {

Purchase\_Order\_c purchaseOrder = new Purchase\_Orderc(Id = purchaseOrderId, Total\_Order\_cost\_c = purchaseToUpdateMap.get(purchaseOrderId));

purchaseToUpdate.add(purchaseOrder);

}

// Update Purchase Orders if there are any changes

if (!purchaseToUpdate.isEmpty()) {

update purchaseToUpdate;

}

     }

    }

}

**GitHub & Project Demo Link :** [**https://github.com/Srividya2315/Medical-Inventory-Management.git**](https://github.com/Srividya2315/Medical-Inventory-Management.git)