**REQUIREMENT ANALYSIS**

|  |  |
| --- | --- |
| Date | 29-05-2025 |
| Team ID | LTVIP2025TMID28829 |
| Project Name | Medical Inventory Management |
| Maximum Marks | 4 Marks |

**3.4 Technology Stack**

**Introduction**

Selecting the right technology stack is crucial for building a secure, scalable, and maintainable Medical Inventory Management System. Given the cloud-native, multi-user, and compliance-sensitive nature of this project, we have adopted Salesforce as the core platform — offering low-code capabilities, enterprise-grade security, and rich integration support.

This section outlines the **complete technology stack** used across **frontend**, **backend**, **automation**, **database**, **security**, and **integration** layers of the solution.

**3.4.1 Platform Overview – Salesforce Ecosystem**

The entire system is implemented on **Salesforce Lightning Platform**, leveraging its cloud infrastructure and modular development capabilities. It offers:

* **Declarative tools** (Process Builder, Flow, Validation Rules)
* **Programmatic extensions** (Apex, Lightning Components, SOQL)
* **Secure cloud hosting** with 99.9% uptime
* **Multi-device support** through responsive Lightning UI

| **Component** | **Technology/Tool** | **Purpose** |
| --- | --- | --- |
| Core Platform | Salesforce Lightning | Base environment for UI, objects, and data |
| Development Language | Apex | Backend logic, triggers, custom classes |
| Declarative Automation | Salesforce Flow | Workflow automation, approval routing |
| User Interface | Lightning App Builder | Visual interface design using components |
| Database Layer | Salesforce Object Model | Custom and standard objects with relationships |
| Query Language | SOQL | Structured queries on Salesforce data |

**3.4.2 Backend Technologies**

Backend logic in Salesforce is built using a combination of **declarative tools** and **Apex code**, depending on complexity.

**Apex**

* Object-oriented, Java-like language used for:
  + Custom validations
  + Scheduled jobs (e.g., expiry reminders)
  + Web service integrations
* Used for advanced logic that can't be achieved with Flows

**Process Automation Tools**

* **Salesforce Flow** (preferred): Used to build logic visually (e.g., auto-create PO on approval)
* **Process Builder**: Used for simple actions like sending notifications or field updates
* **Approval Processes**: Automate multi-level procurement approvals

**Batch Apex & Scheduled Jobs**

* Used for:
  + Expiry notifications
  + Monthly reports
  + Vendor performance analysis

**3.4.3 Frontend & User Experience Layer**

The UI is built using Salesforce Lightning Components and Page Layouts to provide a responsive, intuitive interface for different user roles.

| **Component** | **Technology** | **Purpose** |
| --- | --- | --- |
| UI Framework | Salesforce Lightning Design | Component-based, mobile-friendly front-end |
| Forms & Record Pages | Lightning App Builder | Drag-and-drop page designer for record layouts |
| Dashboards | Lightning Reports & Dashboards | Visualization of KPIs and stock data |
| Mobile Compatibility | Salesforce1 Mobile App | Mobile access without additional development |

**UX Features**

* Color-coded status indicators (low stock, expired)
* Quick action buttons (Add Stock, Raise PO)
* Custom list views with filters for stock and vendors
* Embedded charts in dashboards

**3.4.4 Data Layer & Object Model**

All inventory data, transactions, vendor information, and audit logs are stored using **Salesforce’s object-relational database**.

| **Object Type** | **Examples** |
| --- | --- |
| Standard Objects | User, Profile, Contact, Approval Process |
| Custom Objects | Inventory\_\_c, Vendor\_\_c, Purchase\_Order\_\_c, Stock\_Log\_\_c |
| Relationships | Master-Detail (e.g., Purchase\_Order\_\_c to Item\_\_c) |

**Data Features**

* Field-level security
* Lookup and Master-Detail relationships
* Validation rules (e.g., quantity cannot be negative)
* Record history tracking (auditing)

**3.4.5 Integration & Extensibility**

Salesforce offers rich integration capabilities for connecting external systems such as billing, vendor APIs, or procurement portals.

**Current Integrations**

* Email alerts (standard Salesforce email service)
* CSV imports for bulk data loading

**Future-ready Capabilities**

| **Integration Type** | **Technology** | **Purpose** |
| --- | --- | --- |
| REST API | Apex REST Classes, Named Credentials | Connect with external vendor APIs |
| Data Import | Data Loader, Import Wizard | Import legacy inventory and vendor data |
| 3rd-Party Sync | Salesforce Connect (future use) | Read/write data from external sources |

**3.4.6 Security & Access Control**

Given the medical and compliance sensitivity of the data, the system uses Salesforce’s robust security framework.

| **Security Feature** | **Purpose** |
| --- | --- |
| Profiles & Permission Sets | Define user-level access to objects and fields |
| Role Hierarchies | Ensure proper data visibility by role |
| Record-Level Security | Use Sharing Rules and Org-Wide Defaults |
| Two-Factor Authentication | Protect access to sensitive operations |
| Audit Trail | Log key changes to inventory and user actions |

**3.4.7 Development & Deployment Tools**

| **Tool** | **Purpose** |
| --- | --- |
| Salesforce Dev Console | On-platform development, debug logs, testing |
| Visual Studio Code | Apex development via Salesforce Extensions |
| GitHub (optional) | Version control and team collaboration |
| Salesforce Sandbox | Testing environment before deploying to production |

**Conclusion**

The technology stack selected ensures:

* Seamless user experience across devices
* Efficient backend automation using Salesforce-native tools
* Secure and scalable data management
* Flexibility for future integration with vendor and hospital systems

This stack forms the backbone of a robust and future-ready Medical Inventory Management System that aligns with real-world hospital operations and digital healthcare standards.