Warranty & Service Tracker

Phase 1 : Problem Understanding & Industry Analysis

1. Requirement Gathering

👉 **Goal:** Identify what each stakeholder needs from the system.

Customers (End Users):

- Easy way to register purchased products with warranty details.
- Automatic notifications before warranty expiry.
- Ability to raise and track service requests online.

Service Agents / Technicians:

- Centralized view of all assigned service requests.
- Quick access to product & warranty details for faster issue resolution.
- Reduced manual paperwork.

Warranty Admins / Customer Support Teams:

- Ability to track all warranties (active & expired) in Salesforce.
- Monitor service request lifecycle (new → in progress → closed).
- Automated assignment of requests to the correct service agent.

Management / Executives:

- Dashboards showing warranty coverage and expiry trends.
- Reports on service requests by product model, issue type, and resolution time.
- Data-driven insights to improve product quality and customer retention.

2. Stakeholder Analysis

👉 **Goal:** Define roles and responsibilities.

Primary Stakeholders:

- **Customers** → Register products, receive reminders, create service requests.
- **Service Agents** → Resolve assigned service requests.
- **Warranty Admins** → Manage warranties, reminders, and escalations.

Secondary Stakeholders:

- **Company Management** → Monitor warranty performance and service KPIs.
- **Salesforce Admins / IT Team** → Configure, customize, and maintain the CRM.
- **Third-party Vendors/Partners** → Handle outsourced repairs or escalations (optional).

3. Business Process Mapping

👉 **Goal:** Compare the current manual process vs. Salesforce-enabled solution.

Current Process (Manual/Traditional):

- Customers rely on physical warranty cards and invoices.
- Companies track warranty expiries in Excel or not at all.
- Service requests logged by phone/email → delayed responses.
- No reminders → customers forget to renew or lose coverage.

Proposed Process (Salesforce Enabled):

- **Product & Warranty details stored digitally** in Salesforce.
- **Expiry Date auto-calculated** from Purchase Date + Warranty Term.
- **Automated reminders (Flows/Email Alerts)** sent 30/15/3 days before expiry.
- **Service Requests logged online**, automatically assigned to agents.
- **Dashboards** track active/expired warranties and open/closed service requests.

4. Industry-Specific Use Case Analysis

👉 **Goal:** Benchmark against best practices in warranty & service management.

Warranty Renewal Challenge:

- Customers often miss renewal due to lack of reminders.
- ✅ **Solution:** Salesforce email/SMS reminders before expiry.

Service Request Delays:

- Manual logging causes poor customer satisfaction.
- ✅ **Solution:** Salesforce Service Cloud + Flows to auto-route service cases.

Customer Experience:

- Leading electronics companies provide portals for self-service.

- ✅ **Solution:** Salesforce Experience Cloud portal for product registration & tracking.

Management Visibility:

- Without real-time dashboards, companies lack insights.
- ✅ **Solution:** Salesforce dashboards to monitor warranty/service KPIs.

5. AppExchange Exploration

👉 **Goal:** Identify existing Salesforce apps to accelerate development.

Potential Apps:

- **Service Cloud Extensions** → Advanced service request handling.
- **SMS/Email Reminder Apps** → Automated notifications to customers.
- **Field Service Lightning** → Schedule technicians for repairs.
- **Warranty Management Apps** → Prebuilt solutions for warranty lifecycle management.

6. Standard vs. Custom Objects (Gap Analysis)

👉 **Goal:** Decide which Salesforce objects to reuse and which to build.

- **Account**
  → Reuse for Customer/Company (no changes needed).
- **Contact**
  → Reuse for Individual Customer (no changes needed).
- **Case**
  → Could be reused for Service Requests, but for learning we will create a custom object **Service_Request__c**.
- **Product (Product2)**
  → Could be reused for Products, but it is missing fields like *Serial Number* and *Warranty Term*.
  → We will create a custom **Product__c** object with these fields.