Phase 5: Apex Programming (Developer)

Goal: Add custom logic and automation to enhance Smart Service Request System beyond point-and-click configuration.

1. Apex Triggers

- **Auto-generate Request ID:** A trigger ensures each Service Request gets a unique ID if not already assigned.
- **Validation on Closure:** Prevents a request from being closed unless **Resolution Notes** are entered.
- **Status Updates:** Automatically update the related **Account or Contact** with the latest service request status.

2. Apex Classes

- **Escalation Handler:** A class that checks overdue requests and sends escalation notifications to managers.
- **Assignment Logic:** Assigns incoming service requests to staff based on workload, category, or priority.
- **Utility Methods:** Reusable methods for logging actions, formatting data, or sending custom notifications.

3. Batch Apex

- **Weekly Status Update:** A batch job that reviews all open service requests at the end of the week and sends summary reports to managers.
- **Data Cleanup:** Batch classes can archive closed requests older than a specific time period.

4. Test Classes

- **Unit Tests:** Cover triggers, classes, and batch processes with test data.
- Ensure at least **75% code coverage**, as required by Salesforce for deployment.
- Include **positive, negative, and bulk test cases** to ensure system reliability.

5. Error Handling & Logging

- Use **try-catch blocks** in Apex to handle unexpected errors gracefully.
- Implement a **custom error log object** to capture failures (e.g., assignment errors, escalation failures).
- Send admin alerts when critical errors occur.

6. Future Enhancements

- Add **Queueable Apex** for more complex asynchronous processing.
- Use **Platform Events** for real-time notifications when a high-priority request is escalated.

With Apex programming, Smart Service Request System gains flexibility for complex logic such as automatic assignment, escalations, and reporting, which cannot be achieved with clicks alone.