Phase 9: Testing & Quality Assurance

9.1 Introduction

Testing and Quality Assurance (QA) are critical phases in any Salesforce project to ensure that the solution meets business requirements, functions correctly, and maintains system stability. This phase validates both functional and non-functional requirements before deployment to production.

9.2 Objectives of Testing

- Validate that the Salesforce solution works as expected.
- Identify and fix bugs, errors, and inconsistencies.
- Ensure business requirements are fully implemented.
- Confirm integrations and data flows are reliable.
- Provide stakeholders with confidence in system stability.

9.3 Types of Testing

1. Unit Testing

- Focuses on individual Apex classes, triggers, and Lightning components.
- Developers write test methods to validate business logic.
- Example: Testing Apex triggers for record insert/update events.

2. System Testing

- Ensures all modules (Objects, Flows, Workflows, Validation Rules, etc.) work together.
- Covers end-to-end scenarios like property creation, approval, and updating.

3. Integration Testing

- Validates external APIs, middleware, and third-party connections.
- Example: Testing integration with external property valuation service.

4. User Acceptance Testing (UAT)

- Business users test the system against actual use cases.
- Ensures usability, efficiency, and alignment with business processes.

5. Regression Testing

- Conducted after changes or deployments to ensure existing features remain intact.

9.4 Testing Process

1. **Requirement Review**
    - Review business and technical requirements to define test cases.
2. **Test Planning**
    - Define scope, roles, responsibilities, and timelines.
    - Decide test environments (sandbox vs. production).
3. **Test Case Preparation**
    - Document detailed test cases with inputs, expected results, and actual results.
    - Use tools like Excel sheets, TestRail, or Salesforce's own testing features.
4. **Test Execution**
    - Execute unit, system, and integration test cases.
    - Log defects and issues for resolution.
5. **Defect Tracking & Resolution**
    - Use Jira or Salesforce Cases for tracking.
    - Prioritize and fix issues before production release.
6. **UAT Execution**
    - End users perform acceptance testing.
    - Feedback collected and changes applied.

9.5 Tools & Frameworks

- **Salesforce Apex Test Classes** – mandatory for deployment.
- **Jira / Azure DevOps** – defect tracking and management.
- **Selenium / Provar** – for automated UI testing.
- **Postman** – for API integration testing.
- **Data Loader / Workbench** – for validating bulk data operations.

9.6 Quality Assurance Best Practices

- Maintain at least **75% code coverage** for Apex classes and triggers.
- Use **naming conventions** for test methods to improve readability.
- Always create **positive and negative test scenarios**.
- Involve **business users early** in UAT for better adoption.
- Document all test cases, results, and defects for audit purposes.

9.7 Sample Test Case

Test Case ID	Description	Input	Expected Output	Actual Output	Status
TC-01	Create new Property record	Enter property details	Property saved successfully	As Expected	Pass
TC-02	Validation on Phone field	Enter letters in phone	Error message displayed	As Expected	Pass
TC-03	Approval process trigger on property	Submit property	Sent for approval	As Expected	Pass

9.8 Expected Outcomes

- A reliable, defect-free Salesforce application.
- Verified system performance and integrations.
- Business stakeholders' confidence before go-live.

- Documentation of testing for future reference.

9.9 Conclusion

This phase ensures that the Salesforce implementation is tested thoroughly at different levels, bugs are fixed, and the solution is production-ready. Quality assurance builds trust and confidence in the solution, reducing risks during deployment.