

■ Project Title:

Intelligent Case Routing for Faster Customer Support

■ **Industry:** Customer Support / IT Services

■ **Project Type:** B2C Salesforce Service Cloud Implementation

■ **Target Users:** Support Agents, Support Managers, and Customers

■ Problem Statement:

Many businesses face delays in resolving customer issues because **support cases are not routed efficiently**. Currently, cases are manually assigned or routed using basic rules (like by region or product), which leads to:

- Longer response times
- Uneven workload distribution among support agents
- Decreased customer satisfaction

To address this, the company wants to implement a **Salesforce Service Cloud solution** to:

- Capture support cases from multiple channels (email, web form, chatbot)
- Automate case assignment using rules, queues, and skill-based routing
- Distribute cases evenly based on agent availability and expertise
- Notify customers automatically about status updates
- Provide real-time dashboards for managers to monitor performance

■ Use Cases:

1■■ Case Capture

- Capture cases from multiple channels: email, web form, chatbot
- Automatically create case records with relevant details

2■■ Case Assignment

- Use assignment rules and skill-based routing to assign cases to the most suitable agent
- Consider workload and availability for balanced distribution

3■■ Customer Notifications

- Send automated emails/SMS to customers when case is created, updated, or resolved

4■■ Case Resolution Tracking

- Track SLA compliance, first response time, and resolution time
- Escalate cases if SLA is about to be breached

5■■ Reporting & Dashboards

- Provide dashboards for managers to track team performance
- Monitor open cases, workload distribution, and resolution trends

■ Expected Outcomes:

- 30–40% reduction in average case resolution time
- Improved SLA compliance and faster first responses
- Balanced workload across agents, reducing burnout
- Higher customer satisfaction (CSAT) and retention
- Real-time insights for managers to make data-driven decisions

Phase 2 — Salesforce Org Setup & Configuration

Project: Intelligent Case Routing for Faster Customer Support

This document summarizes the work performed in **Phase 2** of the capstone: setting up the Salesforce Developer Org and configuring the resources required for the *Intelligent Case Routing* project. It includes step-by-step actions completed and a screenshot of the custom object & fields created for routing configuration.

Step 1 — Sign up & Login

Signed up for a Salesforce Developer Edition and logged into the Lightning Experience. Confirmed access to the Setup area using the gear icon (⚙️).

Step 2 — Open Setup & Object Manager

From Setup, opened **Object Manager** to create and manage custom objects. This is where the custom object for routing configuration was created.

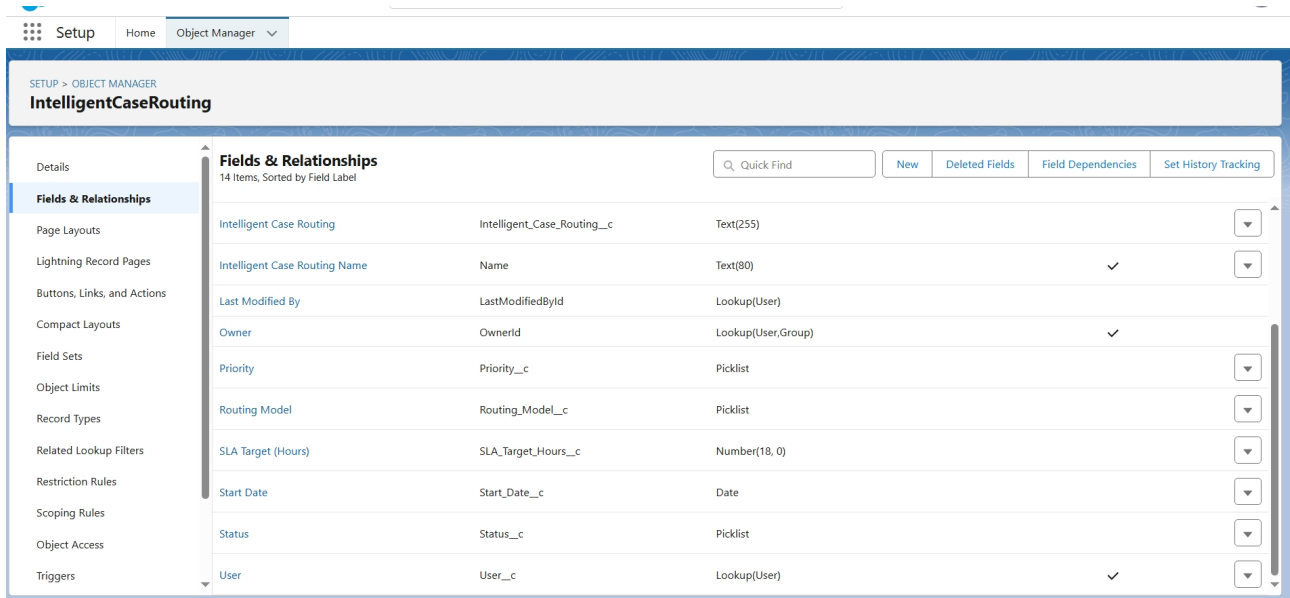
Step 3 — Create Custom Object

Created a custom object to hold routing configurations. Object name shown in Object Manager: **IntelligentCaseRouting** (API name: Intelligent_Case_Routing__c). Enabled 'Allow Reports' and configured record name and description.

Step 4 — Add Fields & Relationships (Key fields added)

Added the most important fields required for intelligent routing and project tracking. The fields created include Project/record identifiers, routing controls, and ownership fields. See the screenshot below showing the Fields & Relationships list for the object.

Screenshot — Object: Fields & Relationships



The screenshot displays the Salesforce Object Manager interface for the 'IntelligentCaseRouting' object. The left sidebar shows navigation options like Details, Fields & Relationships, Page Layouts, and Lightning Record Pages. The main area is titled 'Fields & Relationships' and lists 14 items. The fields are as follows:

Field Label	API Name	Type	Required
Intelligent Case Routing	Intelligent_Case_Routing__c	Text(255)	
Intelligent Case Routing Name	Name	Text(80)	✓
Last Modified By	LastModifiedById	Lookup(User)	
Owner	OwnerId	Lookup(User,Group)	✓
Priority	Priority__c	Picklist	
Routing Model	Routing_Model__c	Picklist	
SLA Target (Hours)	SLA_Target_Hours__c	Number(18, 0)	
Start Date	Start_Date__c	Date	
Status	Status__c	Picklist	
User	User__c	Lookup(User)	✓

Summary of Important Fields Created (from screenshot)

Field Label	API Name	Type
Intelligent Case Routing	Intelligent_Case_Routing__c	Text (255)
Intelligent Case Routing Name	Name	Text (80)

Last Modified By	LastModifiedById	Lookup(User)
Owner	OwnerId	Lookup(User,Group)
Priority	Priority__c	Picklist
Routing Model	Routing_Model__c	Picklist
SLA Target (Hours)	SLA_Target_Hours__c	Number
Start Date	Start_Date__c	Date
Status	Status__c	Picklist
User	User__c	Lookup(User)

Step 5 — Create Custom Tab

Created a Custom Object Tab for the 'Project Details' / 'IntelligentCaseRouting' object so it appears in the App navigation. Selected a tab icon and set default visibility for required profiles.

Step 6 — Add Tab to Lightning App

Opened App Manager → Edit the Lightning App → Navigation Items and added the custom object tab to the selected items so users can access it from the app navigation bar.

Step 7 — Field Level Security & Page Layouts

Configured Field-Level Security for relevant profiles and added fields to the page layout. Ensured managers and admins have visibility and edit rights as needed.

Step 8 — Profiles & Permission Sets

Assigned access to System Administrator and created/used custom profiles or permission sets to grant the required object permissions (Read/Create/Edit). Recommended creating a Support Manager profile or a permission set for managers.

Step 9 — Validation & Testing

Added validation rules and tested record creation. Created sample records to confirm the fields, lookups, and related lists are working as intended.

Step 10 — Documentation & Repository

Captured screenshots (like the one included), documented steps in a README, and uploaded assets to the project repository under a folder such as /Phase2_Salesforce_Setup/.

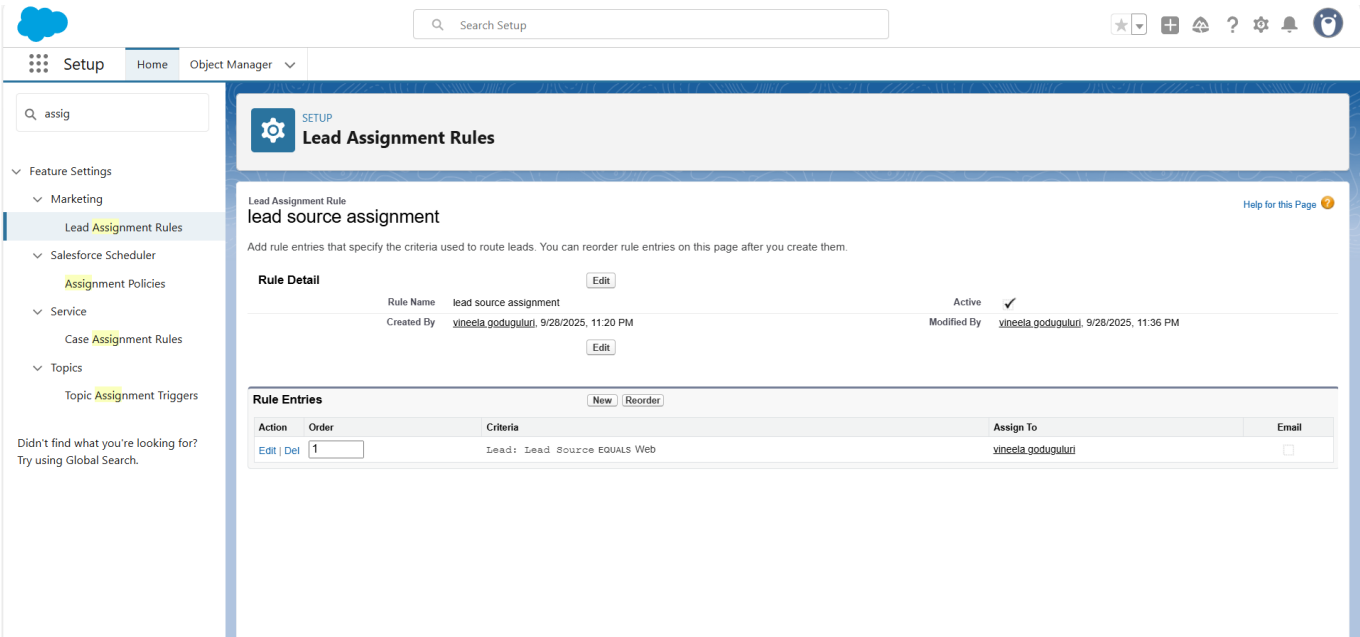
Notes / Recommendations

- Use Permission Sets where possible instead of editing profiles for quick access control.
- Use the custom object to store routing configurations and link cases using a Lookup(Case) if you need to associate projects with cases.
- Build simple reports and a dashboard to monitor SLA breaches, routing performance, and project status.
- Keep screenshots and the step-by-step guide in your repository to support your project submission.

Phase 3 — Step-by-step Implementation (Data Modeling & Relationships)

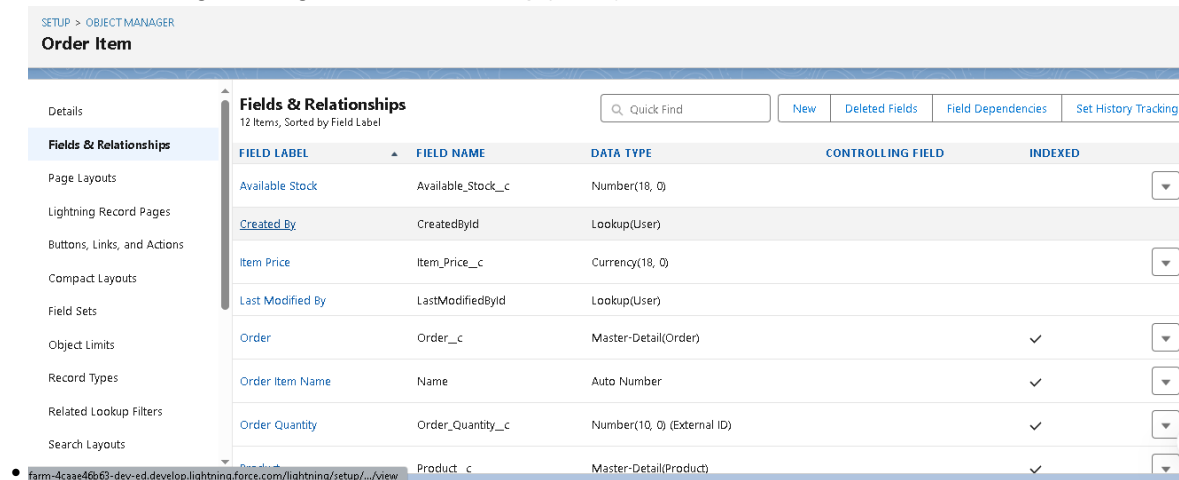
Step 1 — Plan Objects & Fields

- 1.1 Identify standard objects to use: Case, Account, Contact, User, Queue.
- 1.2 Define custom objects: RoutingRule__c, PriorityMatrix__c, Feedback__c.
- 1.3 List required custom fields (Category__c, Severity__c, Assigned_Agent__c, Resolution_Time__c, External_Reference_ID__c).



Step 2 — Create Fields in Setup

- 2.1 Go to Setup → Object Manager → Case → Fields & Relationships → New.
- 2.2 Add Category__c as Picklist (values: Billing, Technical, Account Access, etc.).
- 2.3 Add Severity__c as Picklist (Low, Medium, High, Critical).
- 2.4 Add Assigned_Agent__c as Lookup(User) and Resolution_Time__c as Number.



Step 3 — Configure Record Types

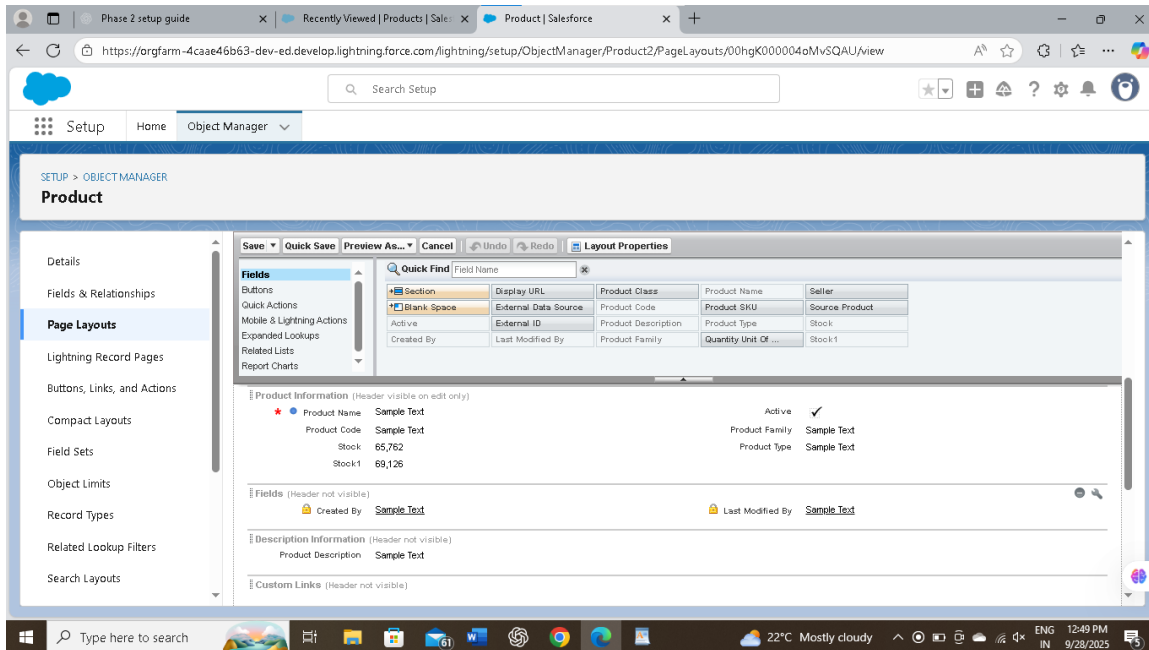
- 3.1 Setup → Object Manager → Case → Record Types → New.
- 3.2 Create 'Customer Support Case', 'Internal IT Request', 'Escalated Case'.
- 3.3 Assign Record Types to profiles (Support Agent, Manager) and set default picklists per type.
-

Step 4 — Customize Page Layouts

- 4.1 Setup → Object Manager → Case → Page Layouts → Edit the main layout.
- 4.2 Add fields: Category__c, Severity__c, Assigned_Agent__c, Resolution_Time__c.
- 4.3 Add Related Lists: Feedback, Routing History, CaseAssignment__c.

Step 5 — Create Compact Layouts

- 5.1 Setup → Object Manager → Case → Compact Layouts → New.
- 5.2 Include: Case Number, Category, Severity, Status, Assigned Agent for mobile highlights.



Step 6 — Define Relationships

- 6.1 Create Master-Detail: Feedback__c → Case (child records deleted with parent).
- 6.2 Create Lookup: Case → User (Assigned_Agent__c), Case → Account, Case → Contact.
- 6.3 Use hierarchical on User to model manager → agent reporting if needed.

Step 7 — Create Junction Object for Flexible Routing

- 7.1 Create CaseAssignment__c with two lookup fields: Case__c and RoutingRule__c.
- 7.2 Add CaseAssignment__c related lists to Case and RoutingRule page layouts.
- 7.3 Use it so a single case can match multiple routing rules and vice versa.

Step 8 — Schema Builder Validation

- 8.1 Open Setup → Schema Builder.
- 8.2 Drag required objects onto canvas and visually confirm relationships.
- 8.3 Save and note any orphaned fields or missing links for cleanup.

Step 9 — External Objects & Integration Mapping (Optional)

- 9.1 Create Named Credential and External Data Source for external ticket/chat platforms.
- 9.2 Expose External_Case_Data__x and map External_Reference_ID__c to external IDs.
- 9.3 Test read-only/external lookups before relying on them in routing logic.

Step 10 — Testing & Sample Data

- 10.1 Create sample Accounts, Contacts, Cases, RoutingRule__c, and CaseAssignment__c records.
- 10.2 Simulate case creation with different Category and Severity values.
- 10.3 Verify case is routed to correct Queue/Assigned_Agent__c per rules.

Step 11 — Data Import & External ID Mapping

- 11.1 Use Data Loader or Import Wizard to bulk import Accounts/Contacts using External_Reference_ID__c as External ID.
- 11.2 Map products or third-party records to Salesforce records using External IDs.
- 11.3 Validate imported records and correct any mapping errors.

Step 12 — Profiles, Permissions & Deployment

- 12.1 Assign Record Types and Page Layouts to profiles (Support Agent, Manager).
- 12.2 Create Permission Sets for access to custom objects/fields as needed.
- 12.3 Deploy from sandbox to production using Change Sets or SFDX; run post-deployment smoke tests.

Step 13 — Monitoring, Reports & Feedback Loop

- 13.1 Build reports: Cases by Queue, Average Resolution_Time__c by Category, RoutingRule matches.
- 13.2 Create dashboards for support managers (backlog, SLA breaches, agent workload).
- 13.3 Use Feedback__c data to refine PriorityMatrix__c and routing criteria iteratively.

Phase 4 — Process Automation (Step-by-step)

Step 1 — Plan Automations & Objectives

- 1.1 Identify key automation goals: faster assignment, SLA enforcement, automatic escalation, and reporting.
- 1.2 Map events that trigger automations: Case creation, status changes, SLA breaches, customer replies.
- 1.3 Choose tools: Record-Triggered Flows (preferred), Assignment Rules, Omni-Channel, Entitlements & Milestones.

Step 2 — Create Validation Rules (Data Integrity)

- 2.1 Setup → Object Manager → Case → Validation Rules → New.

- 2.2 Example: Require Category on new cases:

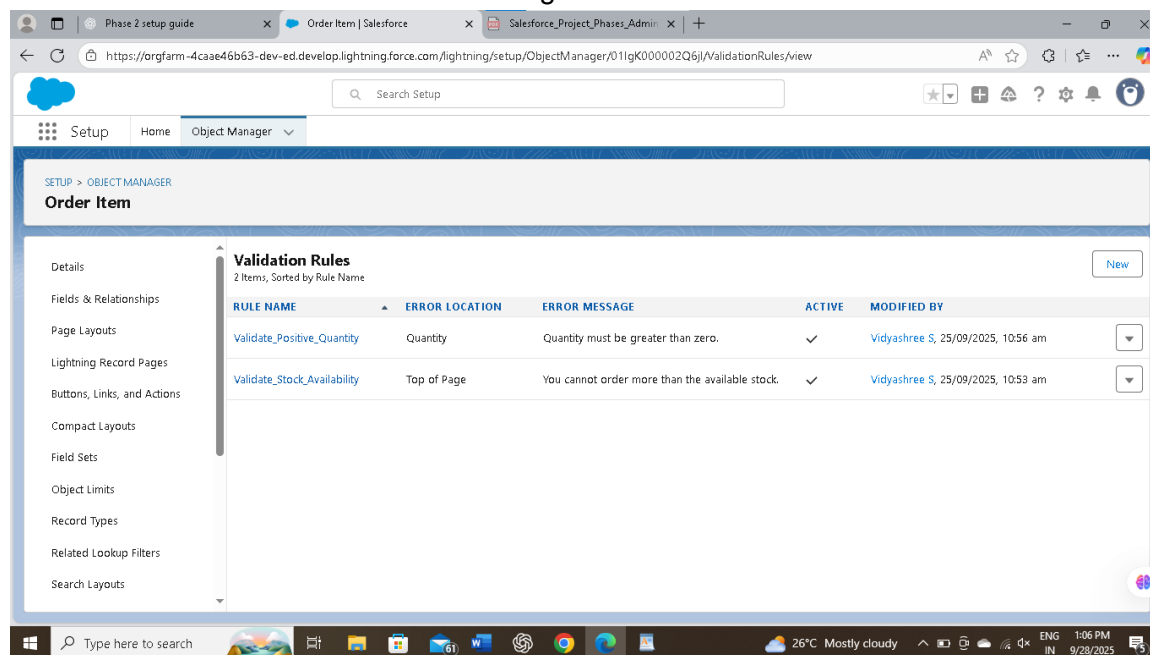
Formula: `AND(ISPICKVAL(Status, 'New'), ISBLANK(TEXT(Category__c)))`

Error: 'Please select a Category for new cases.'

- 2.3 Example: Prevent closing without resolution:

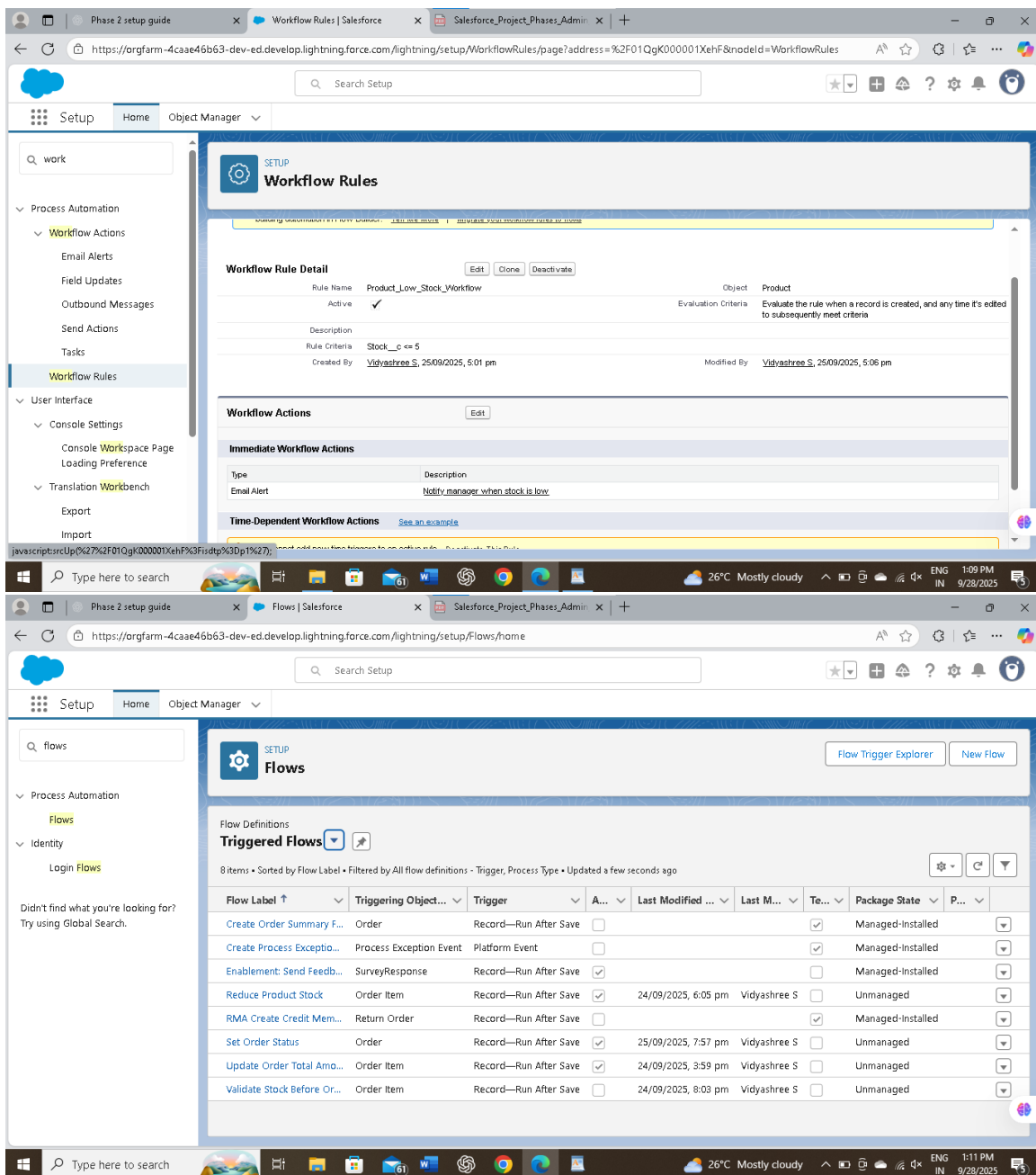
Formula: `AND(ISPICKVAL(Status, 'Closed'), ISBLANK(Resolution_Notes__c))`

Error: 'Add resolution notes before closing this case.'



Step 3 — Build Case Assignment Rules (basic routing)

- 3.1 Setup → Case Assignment Rules → New Rule → Name it (e.g., 'Initial Routing').
- 3.2 Add Rule Entries with order: criteria by Category, Severity, Product, or Account Type.
- 3.3 For each entry, set Assignment To → Queue or User and enable 'Assign using active assignment rule' on case creation where needed.



Step 4 — Set Up Omni Channel & Routing Configs (real-time agent routing)

- 4.1 Setup → Omni-Channel → Service Channels → New (Service Channel = 'Cases').
- 4.2 Setup → Omni-Channel → Routing Configurations → New (set routing model: Least Active, Most Available, or Priority).
- 4.3 Create Presence Statuses & Presence Configurations; set user capacities and assign to agents.
- 4.4 Create Queues (Support Tier 1, Tier 2, Escalation) and associate them with Routing Configurations.

Step 5 — Create Skills for Skills-based Routing (optional)

- 5.1 Setup → Skills → New Skill (e.g., 'Billing', 'Technical', 'Spanish').
- 5.2 Assign Skills to Users (Agent Profiles) and to Cases using a lookup or custom field.
- 5.3 Use Routing Configs that respect Skills for match-based routing via Omni-Channel.

Step 6 — Build Record Triggered Flows for Smart Assignment

- 6.1 Setup → Flows → New → Record-Triggered Flow (Case) → Trigger on Create and Update (as needed).
- 6.2 Add Decision element to evaluate Category, Severity, SLA and custom RoutingRule__c matches.
- 6.3 Use Update Records (Change Owner) to assign OwnerId to User or Queue OR use 'Change Record Owner' action.
- 6.4 Create a CaseAssignment__c record (log) for audit: fields Case__c, RoutingRule__c, AssignedTo__c.
- 6.5 Optionally call 'Send Notification' actions or create Tasks for the newly assigned agent.

Step 7 — Implement SLA Enforcement with Entitlements & Milestones

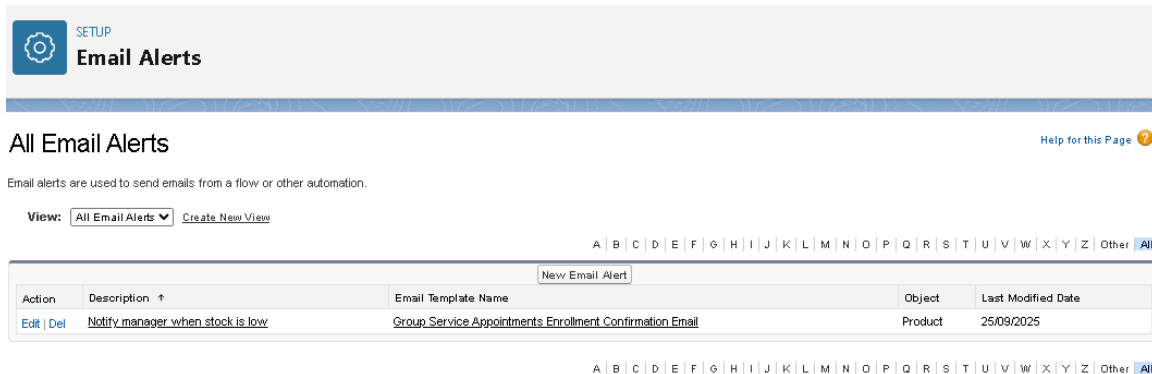
- 7.1 Setup → Entitlements → New Entitlement Process (e.g., 'Standard Support SLA').
- 7.2 Create Milestones: 'First Response' (1 hour), 'Resolve' (72 hours).
- 7.3 Configure Milestone Actions: Email Alert, Post-Update, or Escalate Owner when missed.
- 7.4 Attach Entitlement to Cases automatically via Flow or Process when case meets criteria.

The screenshot shows the Salesforce Setup page for Flows. The left sidebar contains a search bar and a navigation menu with 'Process Automation' and 'Identity' sections. The main content area is titled 'Flows' and shows a list of 8 triggered flows. The table columns are: Flow Label, Triggering Object, Trigger, Action, Last Modified, Last Modified By, Package State, and a dropdown menu. The flows are sorted by Flow Label and filtered by All flow definitions - Trigger, Process Type - Updated a few seconds ago.

Flow Label	Triggering Object	Trigger	Action	Last Modified	Last Modified By	Package State	
Create Order Summary F...	Order	Record—Run After Save	<input type="checkbox"/>			Managed-Installed	
Create Process Exception...	Process Exception Event	Platform Event	<input type="checkbox"/>			Managed-Installed	
Enablement: Send Feedb...	SurveyResponse	Record—Run After Save	<input checked="" type="checkbox"/>			Managed-Installed	
Reduce Product Stock	Order Item	Record—Run After Save	<input checked="" type="checkbox"/>	24/09/2025, 6:05 pm	Vidyashree S	Unmanaged	
RMA Create Credit Mem...	Return Order	Record—Run After Save	<input type="checkbox"/>			Managed-Installed	
Set Order Status	Order	Record—Run After Save	<input checked="" type="checkbox"/>	25/09/2025, 7:57 pm	Vidyashree S	Unmanaged	
Update Order Total Amo...	Order Item	Record—Run After Save	<input checked="" type="checkbox"/>	24/09/2025, 3:59 pm	Vidyashree S	Unmanaged	
Validate Stock Before Or...	Order Item	Record—Run After Save	<input type="checkbox"/>	24/09/2025, 8:03 pm	Vidyashree S	Unmanaged	

Step 8 — Time-Based Flows & Escalation Paths

- 8.1 Use Scheduled Paths in Record-Triggered Flows for time-based checks (e.g., 30 min, 4 hours).
- 8.2 Scheduled Path example: If First Response milestone not completed within 60 minutes → Update Owner to Escalation Queue and send notification.
- 8.3 Alternatively, use Case Escalation Rules (Setup → Case Escalation Rules) for classic time-based escalations.



Step 9 — Approval Process for Special Exceptions

- 9.1 Setup → Approval Processes → Case → New Approval Process (e.g., 'Refund/Escalation Approval').
- 9.2 Define entry criteria (Case Type = 'Refund' OR Severity = 'Critical').
- 9.3 Define Approvers (Manager) and post-approval actions (Change Status to 'Escalated' or assign to specialist).

Step 10 — Notifications, Email Alerts & Templates

- 10.1 Create Email Templates (Classic or Lightning Email Templates) for assignments and SLA breach alerts.
- 10.2 Setup → Email Alerts → New: Tie to Flow or Escalation actions.
- 10.3 Create Custom Notifications (Setup → Notifications) and use Flow action 'Send Custom Notification' for in-app alerts.

Step 11 — Auto-Tasks & Follow-ups

- 11.1 Use Flow to Create Task on case assignment (Subject: 'Acknowledge Case', Due Date: Today + 1 hour).
- 11.2 Auto-create follow-up tasks when milestones are missed to ensure handoff tracking.

Step 12 — Testing (Sandbox) & Test Cases

- 12.1 Create a matrix of test cases: combinations of Category, Severity, Account Type, and Customer SLA.
- 12.2 Test each path: assignment rule, flow decision branch, Omni-Channel routing, scheduled path escalation, and approval flow.
- 12.3 Verify audit logs: CaseAssignment__c entries, Owner changes, email notifications, and milestones triggered.

Step 13 — Reports, Dashboards & Monitoring

- 13.1 Build Reports: 'Unassigned Cases by Queue', 'SLA Breaches', 'Average First Response Time by Agent', 'RoutingRule Matches'.
- 13.2 Create Dashboards for Support Managers to monitor backlog, SLA breaches, and agent load.
- 13.3 Schedule report subscriptions and alerts for SLA breach thresholds

Step 14 — Deploy, Train & Iterate

- 14.1 Migrate Flows, Assignment Rules, Email Templates from Sandbox to Production using Change Sets or SFDX.
- 14.2 Provide agent training documentation and run a pilot with a small team.

- 14.3 Collect feedback (Feedback__c) and iterate on RoutingRule__c and PriorityMatrix__c regularly.

Step 15 — Security, Permissions & Rollback Plan

- 15.1 Ensure profiles and permission sets allow 'Assign Cases', 'Run Flows', 'Edit Case' as required.
- 15.2 Set Flow version management and maintain rollback procedures (deactivate new flow versions if needed).
- 15.3 Keep backups of critical configuration metadata and document changes.

Phase 5 — Apex Programming (Developer) — Step-by-Step

Step 1 — Plan & Scope

- 1.1 Identify scenarios requiring Apex (complex routing decisions, external callouts, bulk recalculation of routing rules, audit logging).
 - 1.2 Map which behaviors remain declarative (Flows, Assignment Rules, Omni-Channel) and which need Apex (bulk callouts, heavy processing, retries).
 - 1.3 Design data contracts: what Apex will read/write (Case fields, RoutingRule__c, CaseAssignment__c, Feedback__c).
-

Step 2 — Developer Environment & Version Control

- 2.1 Install Salesforce CLI + VS Code + Salesforce Extension Pack.
 - 2.2 Create a scratch org or sandbox for development.
 - 2.3 Initialize a Git repo (feature branches) and use SFDX for metadata tracking.
-

Step 3 — Trigger Framework & Design Pattern

- 3.1 Implement “one trigger per object” + separate handler class pattern.
 - 3.2 Create a generic trigger template for Case with delegated handler calls (before/after insert/update/delete).
 - 3.3 Keep trigger bodies minimal — only orchestration.
-

Step 4 — Implement Case Trigger & Handler (basic)

- 4.1 **Before insert / update:** data validation (required Category__c, Severity__c rules).
 - 4.2 **After insert / after update:** evaluate routing (call RoutingService), persist CaseAssignment__c entries for audit, set/ change OwnerId via DML when needed.
 - 4.3 Ensure handler methods accept Lists and Maps (bulkified signatures).
-

Step 5 — Bulkification & Collections Best Practices

- 5.1 Collect IDs and aggregate data outside loops (use Sets for IDs).
 - 5.2 Use Maps for lookup maps (Map<Id, RoutingRule__c>, Map<Id, User>).
 - 5.3 Do not perform SOQL/DML inside loops — perform single queries and batched DML.
-

Step 6 — SOQL & SOSL Hygiene

- 6.1 Query only needed fields.
 - 6.2 Use FOR loops with sub-selects only when efficient.
 - 6.3 Use aggregate queries for counts/metrics where appropriate.
-

Step 7 — Asynchronous Patterns

- 7.1 **Queueable Apex**: for post-assignment processing and callouts (supports chaining).
 - 7.2 **Batch Apex**: for nightly re-evaluation of routing rules across large case sets (use Database.Batchable).
 - 7.3 **Scheduled Apex**: schedule batch or maintenance jobs (SLA health checks, rebalancing workloads).
 - 7.4 Prefer Queueable over @future; use @future only for very small, legacy needs.
-

Step 8 — External Callouts & Integrations

- 8.1 Use Named Credentials and Auth Providers for secure callouts.
 - 8.2 Implement Database.AllowsCallouts in Queueable/Batch if making HTTP requests.
 - 8.3 Use HttpCalloutMock in tests for deterministic behavior.
-

Step 9 — Platform Events / Event-Driven Decoupling (optional but recommended)

- 9.1 Publish a Platform Event when assignment decisions are made (for analytics, downstream sync).
 - 9.2 Create subscribers (Apex Trigger on Platform Event or external system).
 - 9.3 Use events to decouple heavy integrations from synchronous case creation.
-

Step 10 — Logging, Exception Handling & Retries

- 10.1 Wrap callouts and complex logic in try/catch and create Audit__c or Apex_Error__c records for failed flows.
 - 10.2 For transient failures, enqueue a Queueable retry with exponential backoff (store retry count).
 - 10.3 Avoid surfacing raw exception messages to end users — log details and show friendly messages.
-

Step 11 — Security & Sharing

11.1 Use with sharing / without sharing intentionally; prefer with sharing for data-sensitive operations.

11.2 Respect CRUD/FLS — use Schema.sObjectType checks or Security.stripInaccessible as needed.

11.3 Ensure Apex runs with the appropriate user context for assignments.

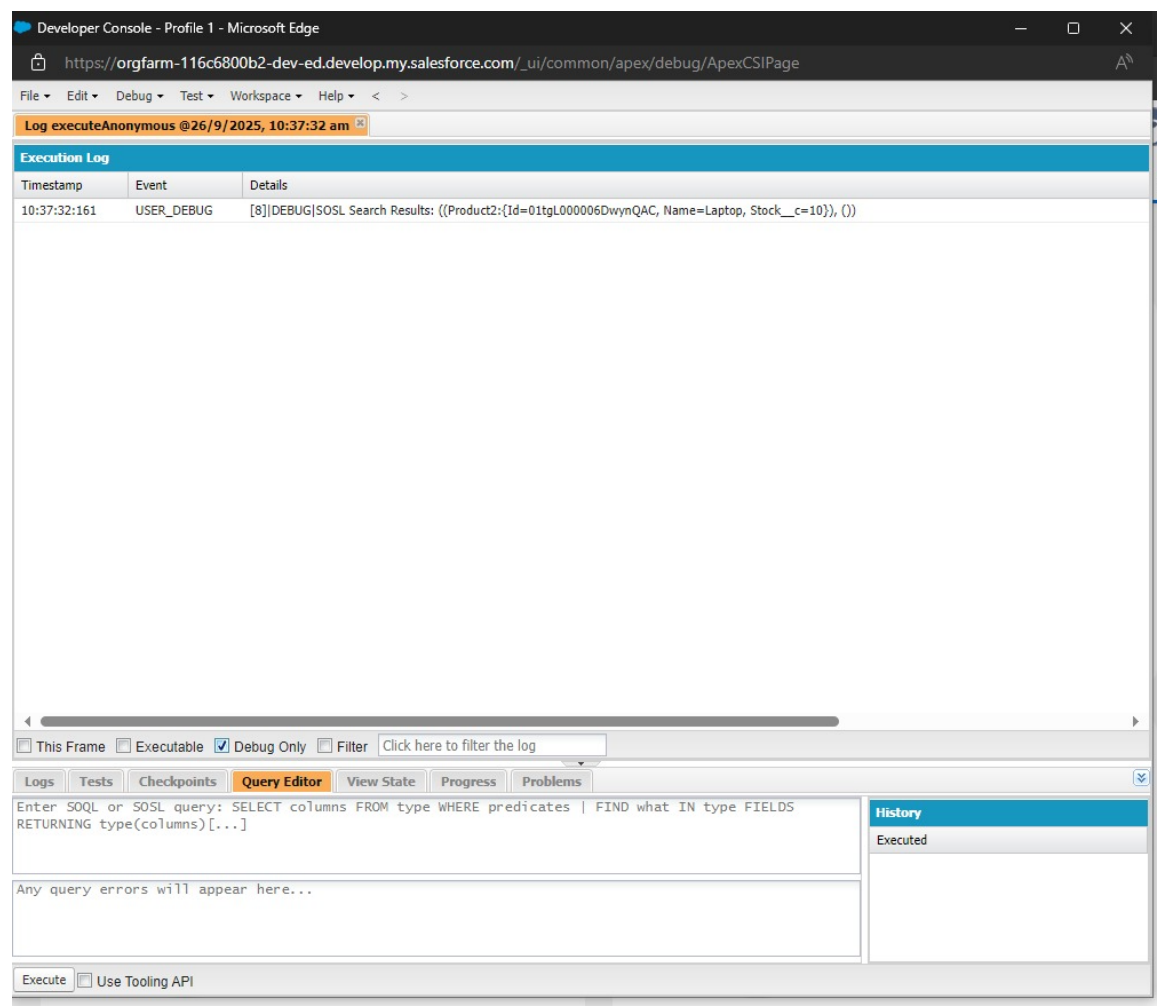
Step 12 — Test Strategy & Quality Gates

12.1 Create @IsTest classes for every class/trigger. Cover happy path, bulk path, negative path, and callout scenarios.

12.2 Use Test.startTest() / Test.stopTest() to simulate async jobs and execute scheduled/batch jobs.

12.3 Mock HTTP callouts with HttpCalloutMock; assert logs and CaseAssignment__c created.

12.4 Maintain code coverage > 75% and assert functional correctness, not only lines covered.

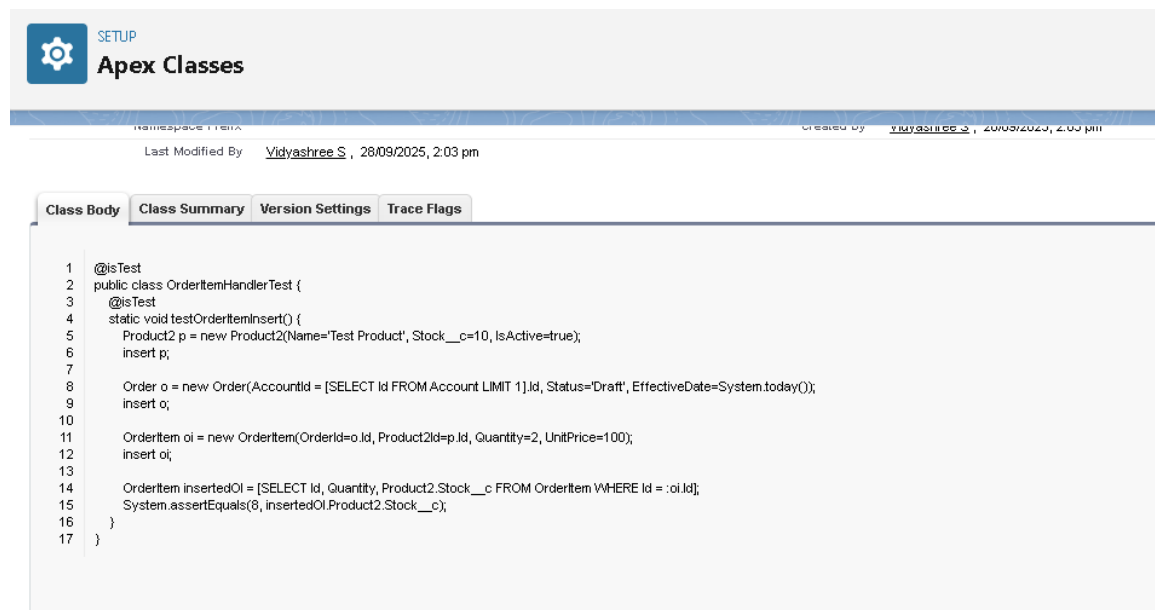


Step 13 — Deployment & CI

- 13.1 Use SFDX or CI (GitHub Actions / Jenkins) to run apex tests on each pull request.
- 13.2 Deploy only validated change sets or SFDX packages; include test run results.
- 13.3 Maintain release rollback steps (deactivate flows, revert apex versions).

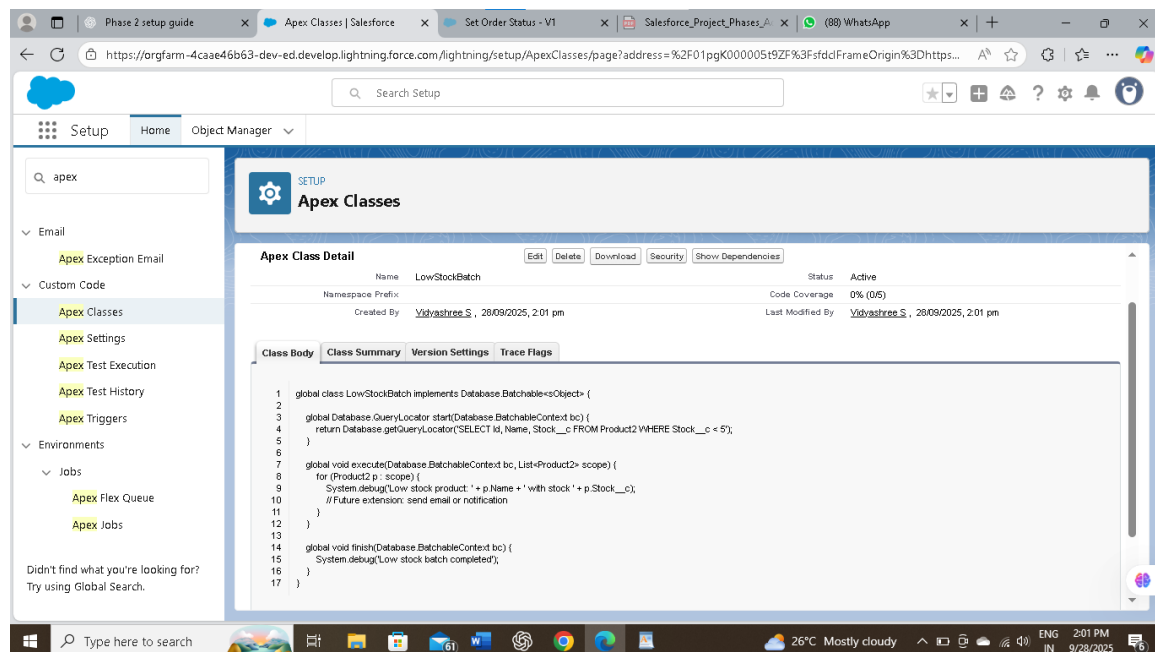
Step 14 — Monitoring & Alerts

- 14.1 Use custom objects (CaseAssignment__c) + reports to monitor assignment rates, failed assignments, and SLA breaches.
- 14.2 Enable Apex Exception Email alerts and set up logging dashboards.
- 14.3 Schedule regular batch jobs to produce summary reports for managers.



The screenshot shows the Salesforce Setup interface for Apex Classes. The page title is "Apex Classes" under the "SETUP" menu. The "Class Body" tab is selected, displaying the following Apex code:

```
1  @isTest
2  public class OrderItemHandlerTest {
3      @isTest
4      static void testOrderItemInsert() {
5          Product2 p = new Product2(Name='Test Product', Stock__c=10, IsActive=true);
6          insert p;
7
8          Order o = new Order(AccountId = [SELECT Id FROM Account LIMIT 1].Id, Status='Draft', EffectiveDate=System.today());
9          insert o;
10
11         OrderItem oi = new OrderItem(OrderId=o.Id, Product2Id=p.Id, Quantity=2, UnitPrice=100);
12         insert oi;
13
14         OrderItem insertedOI = [SELECT Id, Quantity, Product2.Stock__c FROM OrderItem WHERE Id = :oi.Id];
15         System.assertEquals(8, insertedOI.Product2.Stock__c);
16     }
17 }
```



The screenshot shows the Salesforce Setup interface for Apex Classes, displaying the "Apex Class Detail" for the class "LowStockBatch". The class is active and has 0% code coverage. The "Class Body" tab is selected, showing the following Apex code:

```
1  global class LowStockBatch implements Database.Batchable<Object> {
2
3      global Database.QueryLocator start(Database.BatchableContext bc) {
4          return Database.getQueryLocator("SELECT Id, Name, Stock__c FROM Product2 WHERE Stock__c < 5");
5      }
6
7      global void execute(Database.BatchableContext bc, List<Product2> scope) {
8          for (Product2 p : scope) {
9              System.debug('Low stock product: ' + p.Name + ' with stock ' + p.Stock__c);
10             // Future extension: send email or notification
11          }
12      }
13
14      global void finish(Database.BatchableContext bc) {
15          System.debug('Low stock batch completed');
16      }
17 }
```


Quick, copy-ready code examples

A. Case Trigger (one trigger only)

```
trigger CaseTrigger on Case (before insert, before update, after insert, after update) {  
    if (Trigger.isBefore) {  
        if (Trigger.isInsert) CaseTriggerHandler.beforeInsert(Trigger.new);  
        if (Trigger.isUpdate) CaseTriggerHandler.beforeUpdate(Trigger.new,  
Trigger.oldMap);  
    }  
    if (Trigger.isAfter) {  
        if (Trigger.isInsert) CaseTriggerHandler.afterInsert(Trigger.new);  
        if (Trigger.isUpdate) CaseTriggerHandler.afterUpdate(Trigger.newMap);  
    }  
}
```

B. Minimal Handler skeleton

```
public with sharing class CaseTriggerHandler {  
    public static void beforeInsert(List<Case> newCases) {  
        for (Case c : newCases) {  
            if (String.isBlank(c.Category__c)) {  
                c.addError('Please select a Category for this case.');            }  
        }  
    }  
}  
  
    public static void afterInsert(List<Case> newCases) {  
        // collect ids for async processing or immediate assignment  
        List<Id> ids = new List<Id>();  
        for (Case c : newCases) ids.add(c.Id);  
        // enqueue assignment job to keep insert fast and avoid long transactions  
        System.enqueueJob(new CaseAssignmentQueueable(ids));  
    }  
}
```

```
}
```

```
// Implement other lifecycle methods similarly, always bulk-safe
```

```
}
```

C. Queueable assignment job (example)

```
public class CaseAssignmentQueueable implements Queueable,
Database.AllowsCallouts {

    private List<Id> caseIds;

    public CaseAssignmentQueueable(List<Id> ids) { this.caseIds = ids; }

    public void execute(QueueableContext ctx) {

        List<Case> cases = [SELECT Id, Category__c, Severity__c FROM Case
WHERE Id IN :caseIds];

        List<Case> updates = new List<Case>();

        List<CaseAssignment__c> audit = new List<CaseAssignment__c>();

        for (Case c : cases) {

            Id assignee = RoutingService.findAssignee(c); // implement lookup logic in
RoutingService

            if (assignee != null) {

                updates.add(new Case(Id = c.Id, OwnerId = assignee));

                audit.add(new CaseAssignment__c(Case__c = c.Id, AssignedTo__c =
assignee));

            }

        }

        if (!updates.isEmpty()) update updates;

        if (!audit.isEmpty()) insert audit;

    }

}
```

D. Outline of a simple test

```

@Test
private class CaseAssignmentTest {

    @Test static void testQueueableAssignment() {

        // Setup test data

        Account acc = new Account(Name='Tst'); insert acc;

        Case c = new Case(Subject='T', Status='New', AccountId=acc.Id,
Category__c='Billing');

        Test.startTest();

        insert c;

        // execute queued jobs

        Test.stopTest();


        // Assert assignment audit record created or Owner changed

        Integer auditCount = [SELECT COUNT() FROM CaseAssignment__c WHERE
Case__c = :c.Id];

        System.assertEquals(1, auditCount);

    }

}

```

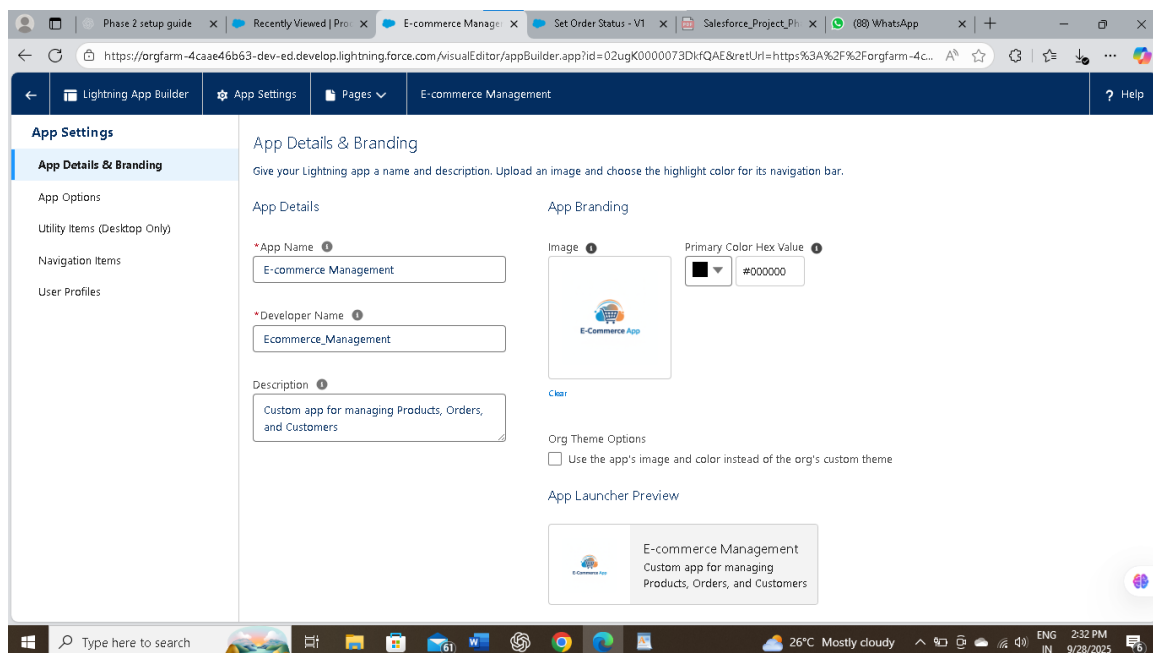
Step-by-Step Process for Intelligent Case Routing for Faster Customer Support

Phase 1 — Goals, scope & KPIs

- Identify channels (email, phone, chat, social, web form, API) and languages to route.
- Define success KPIs: First Response Time (FRT), Mean Time To Resolution (MTTR), SLA breach rate, assignment accuracy, agent utilization, case re-routing rate.
- Identify constraints (PII, regulatory zones, working hours, skills).
- Document stakeholder owners (support managers, data science, IT, legal).
- Deliverables: problem statement, acceptance criteria, placement matrix.

Phase 2 — Data collection & preparation

- Export historical case dataset with fields like subject, description, priority, channel, etc.
- Clean text, redact PII, normalize fields, handle missing labels.
- Label engineering: ensure routing target labels are reliable.
- Create training, validation, and test splits.
- Deliverables: cleaned dataset, data dictionary, baseline routing confusion matrix.



Phase 3 — Design routing strategy (rules + ML hybrid)

- Create deterministic rules for high-certainty cases (VIP, legal, SLA).
- Use ML recommendations for ambiguous cases.
- Define fallback flows for low confidence predictions.
- Set confidence thresholds for auto-assign vs. human review.
- Deliverables: routing decision matrix, thresholds, fallback plan.

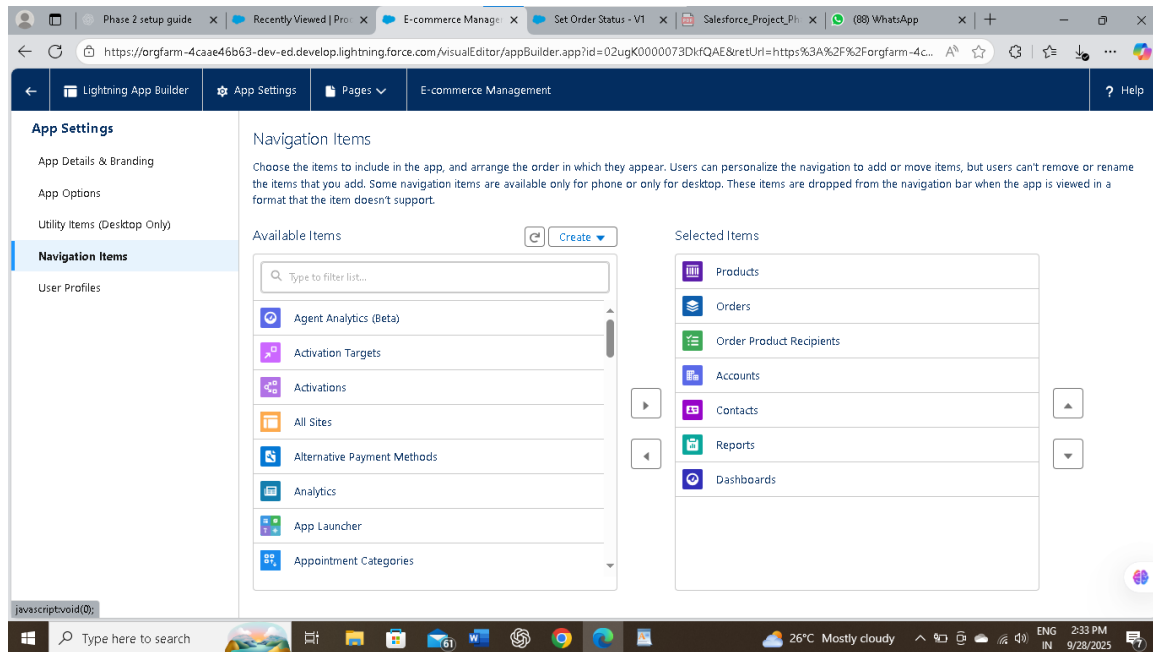
Phase 4 — Build & evaluate ML/NLP models

- Feature set: tokenized subject+description, embeddings, channel, language, product.
- Choose models: XGBoost, LightGBM, or transformer-based text classifiers.
- Monitor metrics: precision, recall, F1, top-1/top-3 accuracy.
- Perform error analysis and human-in-loop validation.

- Deliverables: trained model, evaluation report, sample outputs.

Phase 5 — Salesforce implementation & orchestration

- Configure deterministic rules in Assignment Rules and Flows.
- Use Omni-Channel with Skills-based routing for specialized teams.
- Integrate ML (Einstein Case Classification or external service via Apex/Flow).
- Provide recommendations with confidence scores in UI.
- Deliverables: configured Omni-Channel, Apex/Flow integration, LWC triage component.




Phase 6 — Agent UX & automation

E-commerce Management						
Products ▾ Orders ▾ Accounts ▾ Contacts ▾ Reports ▾ Dashboards ▾						
<div> <div>Products</div> <div>All Products ▾</div> </div> <div> <div>New</div> <div>Add to Category</div> <div>Printable View</div> </div> <div> <div>19 items • Sorted by Product Name • Updated a few seconds ago</div> <div>Search this list...</div> <div> <div>⚙</div> <div>⌵</div> <div>🔄</div> <div>✎</div> <div>🗑</div> <div>⌵</div> </div> </div>						
<input type="checkbox"/>	Product Name ↑	Product Class	Product Code	Product ...	Product ...	
1	<input type="checkbox"/> GenWatt Diesel 1000kW	Simple	GC1060			⌵
2	<input type="checkbox"/> GenWatt Diesel 10kW	Simple	GC1020			⌵
3	<input type="checkbox"/> GenWatt Diesel 200kW	Simple	GC1040			⌵
4	<input type="checkbox"/> GenWatt Gasoline 2000kW	Simple	GC5060			⌵
5	<input type="checkbox"/> GenWatt Gasoline 300kW	Simple	GC5020			⌵
6	<input type="checkbox"/> GenWatt Gasoline 750kW	Simple	GC5040			⌵
7	<input type="checkbox"/> GenWatt Propane 100kW	Simple	GC3020			⌵
8	<input type="checkbox"/> GenWatt Propane 1500kW	Simple	GC3060			⌵
9	<input type="checkbox"/> GenWatt Propane 500kW	Simple	GC3040			⌵
10	<input type="checkbox"/> Installation: Industrial - High	Simple	IN7080			⌵
11	<input type="checkbox"/> Installation: Industrial - Low	Simple	IN7040			⌵
12	<input type="checkbox"/> Installation: Industrial - Medium	Simple	IN7060			⌵

- Design triage Lightning Record Page with recommended queue, confidence, and reasons.
- Enable one-click Accept/Reject/Request Escalation actions.
- Notify via Slack/email for manual triage queues.
- Create macros and quick actions for fast responses.
- Deliverables: LWC Triage Panel, console utilities, macros.

Phase 7 — Testing, pilot & rollout

- Run offline tests with Apex/Flow simulations.
- Pilot with limited case subset or channel.
- Compare KPIs between control and treatment groups.
- Collect agent feedback on routing accuracy and usability.
- Deliverables: pilot performance report, updated rules.

 **Lightning App Builder**

The Lightning App Builder provides an easy to use graphical interface for creating custom Lightning pages for Salesforce Lightning Experience and mobile app. Lightning pages are built using Lightning components—compact, configurable, and reusable elements that you can drag and drop into regions of the page in the Lightning App Builder.

View: All [Create New View](#)

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | **All**

Lightning Pages

New

Action	Label ↑	Name	Namespace Prefix	Description	Type	Created By	Last Modified By
Edit Clone Del	Order Record Page	Order_Record_Page			Record Page	vidya, 24/09/2025, 6:58 pm	vidya, 24/09/2025, 6:58 pm
Edit Clone Del	Order Record Page	Order_Record_Page1			Record Page	vidya, 25/09/2025, 3:43 pm	vidya, 25/09/2025, 3:43 pm
Edit Clone Del	Product Record Page	Product_Record_Page			Record Page	vidya, 25/09/2025, 1:55 pm	vidya, 25/09/2025, 1:55 pm

Intelligent Case Routing for Faster Customer Support

Phase 7: Integration & External Access

- Named Credentials – Secure authentication storage for APIs.
- External Services – Connect Salesforce to external APIs (OpenAPI schema).
- Web Services (REST/SOAP) – Consume or expose Salesforce data.

```
Http http = new Http();
HttpRequest request = new HttpRequest();
request.setEndpoint('callout:FakeStoreEC
/products');
request.setMethod('GET');
HttpResponse response =
http.send(request);
```

- Callouts – Trigger external APIs from Apex or Flows.
- Platform Events – Event-driven routing in real-time.
- Change Data Capture – Keep Salesforce and external data in sync.
- Salesforce Connect – Access external databases without duplication.
- API Limits – Monitor and optimize API consumption.
- OAuth & Authentication – Secure Connected Apps for integrations.
- Remote Site Settings – Whitelist external endpoints for callouts.

The screenshot shows the Salesforce Developer Console interface. At the top, there's a navigation bar with 'File', 'Edit', 'Debug', 'Test', 'Workspace', and 'Help'. Below it, a 'Log' tab is selected, showing an execution log for an anonymous Apex class. The log contains several entries with timestamps, user names, and event details. Below the log, there's a table with columns: User, Application, Operation, Time, Status, Read, and Size. The table shows a single row for a successful execution of an anonymous Apex class.

Timestamp	Event	Details
12:26:29:079	USER_DEBUG	[14]DEBUG:STATUS: Forbidden
12:26:29:079	USER_DEBUG	[11]DEBUG:BODY: <!DOCTYPE html><html lang="en-US"><head><title>Just a moment...</title><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"><meta http-equiv="X-UA-Compatible" content="IE=edge"><meta name="robots" content="noindex,nofollow"><meta name="viewport" content="width=device-width,initial-scale=1"></head><body></body></html>
12:26:29:080	USER_DEBUG	[13]DEBUG:<!DOCTYPE html><html lang="en-US"><head><title>Just a moment...</title><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"><meta http-equiv="X-UA-Compatible" content="IE=edge"><meta name="robots" content="noindex,nofollow"><meta name="viewport" content="width=device-width,initial-scale=1"></body></html>

User	Application	Operation	Time	Status	Read	Size
Udipadine S	Unknown	/services/data/v44.0/tooling/executeanonymous/	9/27/2025, 12:26:29 PM	Success		18.39 KB

Phase 8: Data Management & Deployment

- Data Import Wizard – Simple CSV imports for test/training data.
- Data Loader – Bulk uploads/exports for historical cases.
- Duplicate Rules – Prevent duplicate cases affecting routing.
- Data Export & Backup – Regular exports for disaster recovery.

Data Import Wizard

[Help for this page](#)

Recent Import Jobs

Status	Object	Records Created	Records Updated	Records Failed	Start Date	Processing Time (ms)
Closed	Contact	0	0	0	09-28-2025 03:33	199
Closed	Contact	0	2	0	09-28-2025 03:27	288
Closed	Account	0	0	0	09-28-2025 03:27	214
Closed	Product	2	0	0	09-28-2025 03:23	76
Closed	Contact	2	0	0	09-28-2025 03:11	401

Bulk Api Monitoring

- Change Sets – Deploy routing automation between environments.
- Unmanaged vs Managed Packages – Choose packaging for reuse.
- ANT Migration Tool – Automate metadata deployments.
- VS Code & SFDX – Modern CI/CD and source control integration.

SETUP Duplicate Rules

Account Duplicate Rule Prevent Duplicate Account Names

[Help for this Page](#)

Duplicate Rule Detail

[Edit](#) [Delete](#) [Clone](#) [Activate](#)

Rule Name	Prevent Duplicate Account Names	Order	2 of 2 [Reorder]
Description			
Object	Account		
Record-Level Security	Enforce sharing rules		
Action On Create	Allow	Operations On Create	<input checked="" type="checkbox"/> Alert <input checked="" type="checkbox"/> Report
Action On Edit	Allow	Operations On Edit	<input type="checkbox"/> Alert <input type="checkbox"/> Report
Alert Text	Use one of these records?		
Active	<input type="checkbox"/>		
Matching Rule	<input checked="" type="checkbox"/> Standard Account Matching Rule <input checked="" type="checkbox"/> Mapped	Matching Criteria	Matching rule for account records. More info
Conditions			
Created By	Vidyaashree S, 28/09/2025, 9:07 am	Modified By	Vidyaashree S, 28/09/2025, 9:09 am

[Edit](#) [Delete](#) [Clone](#) [Activate](#)

Phase 9: Reporting, Dashboards & Security Review

- Reports – Tabular, Summary, Matrix, and Joined for routing analysis.
- Report Types – Custom report types for routing metrics.
- Dashboards – Visualize routing performance and SLA compliance.
- Dynamic Dashboards – Personalized dashboards for managers/agents.
- Sharing Settings – Configure case record visibility.

Setup Home Object Manager

Q sha

Security

Guest User Sharing Rule Access Report

Sharing Settings

Didn't find what you're looking for? Try using Global Search.

Sharing Settings

Default Sharing Settings

Organization-Wide Defaults

Object	Default Internal Access	Default External Access	Grant Access Using Hierarchies
Lead	Private	Private	✓
Account and Contract	Private	Private	✓
Contact	Private	Private	✓
Order	Private	Private	✓
Asset	Controlled by Parent	Controlled by Parent	✓
Opportunity	Private	Private	✓
Case	Private	Private	✓
Campaign	Public Full Access	Private	✓
Campaign Member	Controlled by Campaign	Controlled by Campaign	✓
User	Public Read Only	Private	✓
Activity	Private	Private	✓
Calendar	Hide Details and Add Events	Hide Details and Add Events	✓
Price Book	Use	Use	✓
Product	Public Read/Write	Public Read/Write	✓

https://orgfarm-4case46b63-dev-ed.develop.lightning.force.com/lightning/setup/SecuritySharing/home

- Field Level Security – Protect sensitive case fields.
- Session Settings – Enforce secure login/session behavior.
- Login IP Ranges – Restrict access for admins/integration users.
- Audit Trail – Track changes to routing logic and integrations.



Session Settings

Session Settings

Set the session security and session expiration timeout for your organization.

Session Timeout

Timeout Value

2 hours

- ☐ Disable session timeout warning popup
- ☒ Force logout on session timeout

Session Settings

- ☐ Lock sessions to the IP address from which they originated
- ☒ Lock sessions to the domain in which they were first used
- ☐ Terminate all of a user's sessions when an admin resets that user's password [i](#)
- ☒ Force relogin after Login-As-User
- ☐ Require HttpOnly attribute
- ☐ Use POST requests for cross-domain sessions
- ☐ Enforce login IP ranges on every request [i](#)
- ☐ When embedding a Lightning application in a third-party site, use a session token instead of a session cookie.

Extended use of IF11 with Lightning Experience