Phase 7: Integration & External Access

360-Degree Restaurant Lifecycle Management Platform

In Phase 7, the focus of the project was on enabling the restaurant management platform to securely connect with external services and platforms. While the earlier phases concentrated on building core internal features such as order management, kitchen dashboards, and analytics, this phase extended the solution by making it capable of interacting with outside systems for delivery management, payment processing, SMS notifications, and supplier integrations. The integration layer not only ensures smooth data exchange but also improves customer experience and streamlines restaurant operations.

The first major step was the configuration of **Named Credentials**. Named Credentials act as secure storage for authentication details when making external callouts. Four primary integrations were configured: a Delivery Service API for dispatching and tracking orders, a Payment Gateway (Stripe) for handling financial transactions, an SMS Service (Twilio) for customer communications, and a Supplier Portal API for purchase orders and inventory synchronization. Each credential was set up with the appropriate authentication protocol such as OAuth 2.0 for token-based authorization or Password Authentication for services like Twilio. This approach prevents the exposure of sensitive credentials directly in Apex code and ensures reusability across multiple integrations.

Alongside Named Credentials, **Remote Site Settings** were created for whitelisting external domains, which is mandatory for Salesforce outbound callouts. Sites such as api.deliveryservice.com, api.stripe.com, api.twilio.com, and api.supplier-portal.com were registered, along with Google Maps for address validation and delivery routing. This setup ensures that Salesforce trusts these endpoints and allows secure communication.

```
public class DeliveryServiceIntegration {

    private static final String NAMED_CREDENTIAL = 'callout:Delivery_Se

    /**
     * @description Send order to delivery service
     * @param orderId Restaurant Order ID
     * @return String delivery tracking number
     */
    @future(callout=true)
    public static void sendOrderToDeliveryService(Id orderId) {
        try {
            // Get order details
            Restaurant_Order__c order = getOrderDetails(orderId);

            // Prepare request
            HttpRequest req = new HttpRequest();
            req.setEndpoint(NAMED_CREDENTIAL + '/api/v1/orders');
            req.setMethod('POST');
            req.setHeader('Content-Type', 'application/json');
```

The functional integration was then implemented through **Apex REST callouts**. A dedicated Apex class was written for each external service. For the delivery system, the DeliveryServiceIntegration class sends structured JSON payloads containing order details (customer info, items, total, special instructions) and processes responses like

tracking numbers and delivery IDs. It also provides methods for checking delivery status and canceling existing delivery orders. For payment integration, the PaymentGatewayIntegration class connects to Stripe, creating Payment Intents, confirming charges, handling refunds, and retrieving payment details. This ensures end-to-end management of monetary transactions from within Salesforce.

```
public class SMSNotificationService {

    private static final String NAMED_CREDENTIAL = 'callout:SMS_Service';
    private static final String TWILIO_PHONE = '+1234567890'; // Your Twi

    /**
     * @description Send order confirmation SMS
     * @param orderId Order ID
     */
    @future(callout=true)
    public static void sendOrderConfirmationSMS(Id orderId) {
        Restaurant_Order__c order = [
            SELECT Name, Customer__r.Phone, Total_Amount__c, Order_Type__
            FROM Restaurant_Order__c
            WHERE Id = :orderId
        ];

        if (String.isBlank(order.Customer__r.Phone)) {
            System.debug('No phone number for customer');
            return;
        }

        String message = 'Your order #' + order.Name + ' has been confirm
                         'Total: $' + order.Total_Amount__c.setScale(2) +
                         'Type: ' + order.Order_Type__c + '. ' +
                         'We will notify you when ready. Thanks!';
```

For customer communication, the SMSNotificationService was developed using Twilio APIs. It automates the process of sending real-time SMS updates, such as order confirmation, order ready

notifications, and delivery tracking information. It also includes support for promotional campaigns through bulk SMS, improving restaurant–customer engagement. Finally, the SupplierPortalIntegration class was created to handle purchase order submissions, supplier pricing checks, and inventory availability lookups. This ensures that restaurants can seamlessly reorder supplies, verify availability, and keep stock levels updated in real-time.

To complement these API-based interactions, **Platform Events** were introduced to provide asynchronous, event-driven communication. Three events were created: Order_Status_Event__e to notify when order statuses change, Inventory_Alert_Event__e to raise alerts when stock levels are low or items are expiring, and Payment_Event__e to log payment activities. These events decouple the system's components and enable real-time updates across different parts of the application or even external subscribers. A common Apex publisher class, PlatformEventPublisher, was built to standardize how events are published after critical business transactions.

```
public class SupplierPortalIntegration {

    private static final String NAMED_CREDENTIAL = 'callout:Supplier_Porta

    /**
     * @description Send purchase order to supplier
     * @param purchaseOrderId PO ID
     */
    @future(callout=true)
    public static void sendPurchaseOrderToSupplier(Id purchaseOrderId) {
        try {
            Purchase_Order__c po = getPurchaseOrderDetails(purchaseOrderId)

            HttpRequest req = new HttpRequest();
            req.setEndpoint(NAMED_CREDENTIAL + '/api/orders');
            req.setMethod('POST');
            req.setHeader('Content-Type', 'application/json');

            Map<String, Object> poData = new Map<String, Object>{
                'po_number' => po.Name,
                'supplier_id' => po.Supplier__r.External_Supplier_Id__c,
                'requested_delivery_date' => po.Expected_Delivery_Date__c.
                'total_amount' => po.Total_Amount__c,
                'line_items' => buildPOLineItems(po)
            };
```

Overall, Phase 7 extended the application beyond its internal boundaries to interact with the external ecosystem. By leveraging Named Credentials, Remote Site Settings, REST callouts, and Platform Events, the system now supports secure API-driven integrations with delivery providers, payment gateways, SMS services, and supplier networks. This integration layer not only enhances automation but also improves reliability, scalability, and user experience. With this foundation, the restaurant platform can serve as a central hub that coordinates both internal operations and external service providers in a seamless manner.