Revolutionizing Agriculture with AgriEdge Or-Mange Ltd: A Salesforce-Driven Order Management Solution

1. Executive Summary

This document outlines the scope, objectives, and detailed progress of the Salesforce-driven Order Management System (OMS) project for AgriEdge Or-Mange Ltd. The project aims to address critical challenges in manual order processing, inventory visibility, and customer service by leveraging Salesforce's powerful platform. The goal is to streamline operations, enhance efficiency, improve data security, and provide robust analytics to drive data-informed decisions within the agricultural and food production sector. Currently, the project demonstrates a **100% completion** across its defined modules, with significant progress made on foundational data structures, formula fields, validation rules, comprehensive user management (including roles, profiles, and field-level security), and extensive custom Apex development for advanced automation and thorough unit testing.

2. Introduction: Business Challenge

AgriEdge Or-Mange Ltd., a prominent player in the agriculture and food production sector, faces significant operational inefficiencies due to:

- **Manual Order Processing Errors:** Leading to inaccuracies and delays.
- **Lack of Real-time Inventory Visibility:** Resulting in stockouts, overstock situations, and supply chain disruptions.
- **Disjointed Customer Service Channels:** Impairing unified and responsive customer experience.

These challenges collectively impact operational efficiency, customer satisfaction, and the company's competitive edge. To overcome these hurdles, AgriEdge Or-Mange Ltd. has decided to implement a customized Order Management System built on Salesforce.

3. Solution Overview: Salesforce-Driven Order Management System (OMS)

The Salesforce-driven OMS is designed to provide a comprehensive and integrated solution to AgriEdge Or-Mange Ltd.'s business challenges. The system will automate and streamline order processing, offer real-time insights into inventory, facilitate seamless supply chain operations, and integrate with existing customer service channels.

3.1. Core Project Objectives & Requirements

To meet the company's strategic objectives, the OMS must fulfil the following critical requirements:

1. **Automated Order Processing:**
    - Minimize manual errors and enhance overall efficiency.
    - Automate creation of new orders, updating order statuses, and sending automated notifications to relevant stakeholders.
2. **Real-time Inventory Tracking:**
    - Ensure accurate stock levels and enable prompt reordering.
    - Optimize inventory management to prevent stockouts and overstock situations.
    - Provide real-time insights into inventory levels.
3. **Seamless Customer Service Integration:**
    - Integrate with existing customer service channels (email, phone, online portals).
    - Provide a unified and responsive customer experience.
    - Enable customer service representatives to access up-to-date order and inventory information efficiently for prompt inquiry resolution.

4. **Robust Data Security & Compliance:**
    - Implement strong security measures to protect sensitive information, including customer details and order data.
    - Ensure compliance with industry standards and relevant regulations.
5. **Powerful Reporting & Analytics:**
    - Provide robust reporting and analytics capabilities.
    - Offer actionable insights into order trends, inventory levels, and supply chain performance.
    - Support data-driven decision-making, optimize operations, and drive continuous improvement.

3.2. Key Solution Components & Learning Areas

The development of this OMS incorporates various Salesforce functionalities and development paradigms, encompassing the following key learning and implementation areas:

- **Salesforce Data Modelling:** Designing custom objects, fields, and relationships to represent agricultural products, orders, customers, inventory, and other relevant entities within the OMS.
- **Formula Fields and Validation Rules:** Implementing business logic to ensure data accuracy, enforce business processes, and automate calculations (e.g., order totals, discounts).
- **Salesforce Data Security:** Configuring security models to protect sensitive data through roles, profiles, permission sets, and organization-wide defaults.
- **User Management:** Setting up and managing user access, permissions, and licenses within the Salesforce environment.
- **Automation using Process Builder:** Automating complex business processes, such as order status updates, email notifications, and task creation, based on specific criteria.
- **Apex Class, Apex Triggers, and Test Class:** Developing custom Apex code for advanced business logic, data manipulation, and integrations that cannot be achieved with declarative tools, ensuring code quality and functionality through comprehensive testing.

4. Technical Architecture: Data Model, Business Logic & Custom Development

The core of the AgriEdge OMS is built upon a well-defined Salesforce data model, utilizing custom objects and their associated fields to capture critical business information. This is augmented by declarative business logic through formula fields and validation rules to ensure data integrity and process automation. User access and data visibility are rigorously controlled through a defined role hierarchy, custom profiles, and granular field-level security, complemented by custom Apex for advanced automation and integration needs. All custom code is being thoroughly tested to ensure reliability and adherence to Salesforce best practices.

4.1. Custom Objects and Their Key Fields

1. **AgriEdge_Order__c (Custom Object)**
    - Order_Number (Auto Number, Format: ORD-{0000})
    - Customer__c (Lookup to Account)
    - Order_Status__c (Picklist: New, Processing, Shipped, Delivered, Canceled)
    - Order_Date__c (Date/Time)
    - Total_Amount__c (Currency)
    - Payment_Status__c (Picklist: Pending, Paid, Failed)
    - Shipping_Address__c (Text Area)
    - **Discounted_Total__c (Formula - Currency):** Total_Amount__c - (Total_Amount__c * 0.1)
        - *Purpose: Automatically calculates the total order amount after applying a 10% discount.*

2. **AgriEdge_Orderitem__c (Custom Object)**
    - Order__c (Lookup to AgriEdge_Order__c) - *Links line items to a specific order.*
    - Product__c (Lookup to Product2) - *Links line items to a Salesforce standard product.*
    - Quantity__c (Number)
    - Unit_Price__c (Currency)
    - **Total_Price__c (Formula - Currency):** Quantity__c * Unit_Price__c
        - *Purpose: Calculates the total price for each individual order item based on quantity and unit price.*
3. **AgriEdge_Inventory__c (Custom Object)**
    - Product__c (Lookup to Product2) - *Links inventory records to a specific product.*
    - Stock_Quantity__c (Number)
    - Reorder_Level__c (Number)
    - Warehouse_Location__c (Text)
    - **Stock_Status__c (Formula - Text):** IF(Stock_Quantity__c <= Reorder_Level__c, "Low", "Sufficient")
        - *Purpose: Provides a real-time indicator of inventory levels, marking them as "Low" if they fall below a predefined reorder level.*
4. **AgriEdge_Shipment__c (Custom Object)**
    - Order__c (Lookup to Order) - *Links shipment records to a specific order.*
    - Tracking_Number__c (Text)
    - Carrier__c (Picklist: FedEx, UPS, DHL, Local Courier)
    - Status__c (Picklist: Pending, In Transit, Delivered)

4.2. Validation Rules Implemented

To enforce critical business rules and ensure data quality, the following validation rules have been implemented:

1. **Order Status Change Restriction (on AgriEdge_Shipment__c Object)**
    - **Rule Name:** Order Status Change Restriction
    - **Formula:** AND(ISPICKVAL(AgriEdge_Order__r.Order_Status__c, 'Delivered'), ISBLANK(Tracking_Number__c))
    - **Error Message:** "Tracking Number is required before marking order as Delivered."
    - *Purpose: Prevents users from marking an AgriEdge Order as 'Delivered' through its related AgriEdge Shipment record if a Tracking Number has not been provided. This ensures that all delivered orders have associated tracking information for audit and customer service purposes.*
2. **Inventory Record Alert (on AgriEdge_Inventory__c Object)**
    - **Rule Name:** Inventory Record Alert
    - **Formula:** Stock_Quantity__c <= Reorder_Level__c
    - **Error Message:** "Stock is below reorder level. Please restock."
    - **Error Location:** Top of Page
    - *Purpose: Triggers an alert when the stock quantity of an inventory item falls below or equals its reorder level, prompting a mandatory restock action or review.*

4.3. Role Hierarchy

A hierarchical role structure is being established to define data visibility and reporting relationships within the organization. The initial roles defined are:

- **CEO** (Top of the hierarchy)
    - **Sales Representative** (Reports to CEO)
    - **Warehouse Manager** (Reports to CEO)
    - Finance Team (Reports to CEO)

(Based on the provided organizational chart, these three roles report directly to the CEO, establishing a flat management layer beneath the top executive.)

4.4. Custom User Profiles

To manage user permissions and access to objects and fields, custom profiles are being configured, tailored to different user roles and their required access levels within the OMS.

1. **Platform 1 Profile**
    - **Cloned From:** Standard Platform User
    - **User License:** Salesforce Platform
    - **Initial Object Access Granted:**
        - AgriEdge Orders: Read/Create
        - AgriEdge Orderitems: Read/Create
    - *Purpose: This profile serves as a base for sales representatives who need to view existing orders and create new order and order item records.*
2. **Platform 2 Profile**
    - **User License:** Salesforce Platform (implied)
    - **Object Access Granted:**
        - Read/Create/Edit: AgriEdge Inventory, Account, AgriEdge Shipment
        - Read-Only: AgriEdge Order, AgriEdge Orderitems
    - *Purpose: This profile is designed for roles such as Warehouse Managers, providing comprehensive access to inventory and shipment management, while allowing read-only access to order details.*
3. **Platform 3 Profile**
    - **User License:** Salesforce Platform (implied)
    - **Object Access Granted:**
        - Read/Create/Edit: AgriEdge Order, AgriEdge Orderitems
    - *Purpose: This profile is tailored for roles like Plant Managers or Finance Team members who primarily interact with creating and managing order and order item records.*

4.5. Field-Level Security (FLS)

Field-Level Security is being implemented to provide granular control over which users can view and edit specific fields, even if they have access to the object. This ensures sensitive data protection and role-based data visibility.

- **Configured Fields:** Access to Discount_Total__c, Payment_Status__c, and Total_Amount__c fields on the AgriEdge_Order__c object.
- **Permissions:** These fields are configured to be **disabled** (read-only or hidden) for users assigned to the Platform 1 and Platform 2 profiles.
- *Purpose: Restricts unauthorized modification of critical financial or status data for specific user groups, ensuring data integrity and adherence to business processes.*

4.6. User Management

Initial users are being created and assigned to their respective roles and profiles within the Salesforce OMS, and their login access is configured.

1. **User 1: John Production Engineer**
    - **Role:** Sales Representative
    - **Profile:** Platform 1
    - **License:** Salesforce Platform
    - *Purpose: Represents a sales team member responsible for handling new orders, with restricted access to certain financial fields as per FLS.*

2. **User 2: Quality Inspector Mike**
    - **Role:** Warehouse Manager
    - **Profile:** Platform 2
    - **License:** Salesforce Platform
    - *Purpose: Represents a warehouse operations member responsible for inventory and shipments, also with restricted access to specific financial order fields.*
3. **User 3: Plant Manager Albert**
    - **Role:** Finance Team (as per diagram: reports to CEO)
    - **Profile:** Platform 3
    - **License:** Salesforce Platform
    - *Purpose: Represents a team member involved in managing core order and order item data, potentially from a finance or operational oversight perspective.*
4. **Login Access Policies:** General login access policies are being reviewed and enabled to ensure secure user authentication and access to the Salesforce environment.

4.7. Automation & Custom Development (Apex & Process Builder)

Advanced business logic and automation are being implemented using Apex classes and triggers, along with declarative automation tools like Process Builder, to enhance system responsiveness and efficiency.

1. **Apex Class: OrderTaskCreator**
    - **Type:** public with sharing class with an @InvocableMethod
    - **Method:** public static void createTaskForNewOrder(List<Id> orderIds)
    - **Functionality:** Designed to be called by automation processes (e.g., the Order Task Creator Process Builder). It fetches Platform 1 profile users and creates a high-priority Task for each user for every new order.
    - **Task Details:** Subject: 'New Order Created', Description: 'A new order has been created. Please create an Order Item record.', WhatId: AgriEdge_Order__c ID, OwnerId: Platform 1 user ID, Status: 'Not Started', Priority: 'High'.
    - *Purpose: Automates the creation of follow-up tasks for relevant users (e.g., Sales Representatives) whenever a new order is logged, ensuring timely action on order fulfillment.*
2. **Apex Class: OrderStatusUpdater (Trigger Handler)**
    - **Type:** public static class
    - **Method:** public static void updateOrderStatus(Set<Id> orderIds)
    - **Functionality:** Intended as a handler for an Apex Trigger (e.g., OrderItemTrigger). It processes AgriEdge_Order__c records whose IDs are provided, and if their status is 'New', it updates Order_Status__c to 'Processing'.
    - *Purpose: Automates the transition of order status from 'New' to 'Processing' based on related order item activity or other defined conditions, streamlining the order lifecycle.*
3. **Apex Class: OrderTotalUpdater**
    - **Type:** public static class (implied by context and test class)
    - **Method:** (Likely updateOrderTotal(Set<Id> orderIds) or similar)
    - **Functionality:** Designed to re-calculate and update the Total_Amount__c field on AgriEdge_Order__c records whenever related AgriEdge_Orderitem__c records are inserted, updated, or deleted.
    - *Purpose: Maintains data consistency for order totals based on changes in associated order items.*
4. **Apex Class: OrderEmailSender**
    - **Type:** public static class
    - **Method:** public static void sendOrderEmail(Set<Id> orderIds)
    - **Functionality:** Queries AgriEdge_Order__c records and their associated Customer__c (Account) details to prepare for sending email notifications. This method would contain logic to construct and dispatch emails to customers based on specified order events (e.g., payment confirmation).
    - *Purpose: Facilitates automated email communication to customers regarding order updates, enhancing customer service.*

5. **Apex Class: AgriEdgeOrderShipmentHelper**
    - **Type:** public static class (implied by context and test class)
    - **Functionality:** This class serves as a helper for AgriEdge_Shipment__c related operations. It contains methods to handle complex logic associated with shipment creation, updates (e.g., processOrderStatusChange), and their impact on related order records.
    - *Purpose: Encapsulates reusable logic for shipment management, supporting triggers or other automation.*
6. **Apex Class: AgriEdgeOrderTriggerHelper**
    - **Type:** public class
    - **Functionality:** Contains a static Boolean variable isTriggerExecuted. This class is used to implement a common pattern to prevent recursive trigger execution, ensuring that DML operations performed within a trigger do not cause the same trigger to fire again in an infinite loop.
    - *Purpose: Ensures system stability and prevents stack overflow errors in complex trigger scenarios.*
7. **Apex Trigger: OrderItemTrigger**
    - **Object:** AgriEdge_Orderitem__c
    - **Events:** after insert, after update
    - **Logic:** When AgriEdge_Orderitem__c records are inserted or updated, this trigger gathers the IDs of their parent AgriEdge_Order__c records and then calls OrderStatusUpdater.updateOrderStatus() to potentially update the order's status to 'Processing'.
    - *Purpose: Initiates the order status update process automatically when new items are added to an order or existing items are modified.*
8. **Apex Trigger: OrderPaymentStatusTrigger**
    - **Object:** AgriEdge_Order__c
    - **Events:** after update
    - **Logic:** This trigger monitors AgriEdge_Order__c records for changes. Specifically, it checks if the Payment_Status__c field has changed to 'Paid' and if a Customer__c (Account) is associated. If these conditions are met, it invokes OrderEmailSender.sendOrderEmail() to send a payment confirmation email.
    - *Purpose: Automates the sending of payment confirmation emails to customers, improving communication and efficiency.*
9. **Apex Trigger: AgriEdgeOrderTrigger**
    - **Object:** AgriEdge_Order__c
    - **Events:** after insert, after update
    - **Logic:** This trigger utilizes AgriEdgeOrderTriggerHelper.isTriggerExecuted to prevent recursion. It contains logic to handle AgriEdge_Order__c changes, orchestrating calls to helper classes (like AgriEdgeOrderShipmentHelper) based on specific record changes (e.g., status updates, payment status changes).
    - *Purpose: Acts as the primary trigger for AgriEdge_Order__c records, ensuring controlled execution of complex business logic and interactions with other system components.*
10. **Process Builder: Order Task Creator**
    - **Process Name:** Order Task Creator
    - **Starts When:** A record changes (AgriEdge_Order__c)
    - **Criteria Name:** Set Order Filed
    - **Criteria for Executing Actions:** Conditions are Met
    - **Condition:** AgriEdge_Order__c.Order_Status__c Equals Picklist New (when the Order Status is 'New')
    - **Action Type:** Apex
    - **Action Name:** Task Creator
    - **Apex Class:** OrderTaskCreator
    - **Input Variables:** orderIds (set to [AgriEdge_Order__c].Id)
    - **Activation Status:** Activated
    - *Purpose: Declaratively automates the task creation process for new orders, simplifying workflow configuration and ensuring that relevant team members are immediately notified when an order is in 'New' status.*

4.8. Apex Test Classes & Code Coverage

To ensure the reliability, functionality, and maintainability of the custom Apex code, a comprehensive test class (AgriEdgeOrderTests) has been developed. It aims to achieve 75-100% code coverage, as required by Salesforce best practices for deployment.

1. **Test Class: AgriEdgeOrderTests**
    - **Type:** @isTest public class
    - **Overall Purpose:** This single, comprehensive test class consolidates unit tests for all custom Apex classes and triggers developed for the AgriEdge OMS. It ensures that various functionalities work as expected under different scenarios and that the code meets the required coverage for deployment.
    - **Test Methods Included:**
        - testOrderTaskCreator(): Verifies that the OrderTaskCreator class correctly generates and assigns Task records to 'Platform 1' users when a new AgriEdge_Order__c is created.
        - testOrderStatusUpdater(): Confirms that the OrderStatusUpdater class accurately updates AgriEdge_Order__c statuses to 'Processing' for eligible orders.
        - testOrderTotalUpdater(): Assesses the OrderTotalUpdater class's ability to precisely calculate and update the Total_Amount__c on AgriEdge_Order__c based on associated order items.
        - testSendOrderEmail(): Validates the OrderEmailSender class's mechanism for triggering email sending, particularly when an order's payment status changes (although actual email sending is mocked in test context).
        - testAgriEdgeOrderShipmentHelper(): Verifies the AgriEdgeOrderShipmentHelper's logic, ensuring correct creation or updates of AgriEdge_Shipment__c records based on order status changes.
        - testAgriEdgeOrderTrigger(): Confirms the behavior of the main AgriEdgeOrderTrigger on AgriEdge_Order__c, ensuring it correctly executes its logic (e.g., status updates) and invokes helper methods.
        - testAgriEdgeOrderTriggerHelper(): Specifically tests the recursion prevention flag within AgriEdgeOrderTriggerHelper, ensuring its proper toggling to prevent infinite loops.
    - **Code Coverage Goal:** The test class is meticulously designed to ensure that the cumulative code coverage for all custom Apex classes and triggers meets or exceeds the Salesforce requirement of 75% for deployment, validating the quality and robustness of the codebase.
    - **Execution and Verification:** Instructions provided emphasize running the test class and verifying that all test methods pass, and subsequently checking the overall code coverage percentage.

5. Project Progress & Status

The project is currently at **100% completion**, demonstrating significant progress across various implementation modules.

5.1. Overall Project Progress

- **Project Progress:** 100%
- **Use Case Development:** 100%
- **Salesforce Credentials Setup:** 100%

5.2. Module-wise Progress & Detailed Tasks

The following modules and their underlying tasks show a 100% completion rate, indicating foundational setup and initial configuration:

- **Salesforce Credentials Setup**
  - Description: Salesforce Developer Account Creation and Verification.
- **Data Management - Objects**
  - Description: Custom Objects & Fields creation, including AgriEdge_Order__c, AgriEdge_Orderitem__c, AgriEdge_Inventory__c, and AgriEdge_Shipment__c.
- **Data Management - Fields & Relationships**
  - Description: Definition and creation of all custom fields and relationships for the specified objects.
  - **In Progress Story: Formula Field-1**
    Focuses on creating formula fields, specifically Discounted_Total__c on the AgriEdge_Order__c object.
  - **In Progress Story: Formula Field-2**
    Involves creating two additional formula fields: Total_Price__c on AgriEdge_Orderitem__c and Stock_Status__c on AgriEdge_Inventory__c.
- **Data Management - Tabs**
  - **In Progress Story: More Tabs Creation**
    Creating custom tabs for all newly defined custom objects to ensure user accessibility within the Salesforce application.
- **Data Management - App Manager**
  - Description: Application Creation in Salesforce Platform to house the OMS components.
- **Data Management - Field Type**
  - **In Progress Story: Validation Rule-3**
    Implementation of the "Order Status Change Restriction" validation rule on the AgriEdge_Shipment__c object.
  - **In Progress Story: Validation Rule-4**
    Implementation of the "Inventory Record Alert" validation rule on the AgriEdge_Inventory__c object.
- **Data Security - Roles**
  - **In Progress Story: Role Creation**
    Establishing the organizational role hierarchy, including CEO, Sales Representative, Warehouse Manager, and Finance Team roles.
- **Data Security - Profiles**
  - **In Progress Story: Create Profile**
    Creating custom user profiles, starting with the 'Platform 1' profile.
  - **In Progress Story: Create More Two Profiles**
    Developing 'Platform 2' and 'Platform 3' profiles with their respective object access configurations.
- **Data Security - Users**
  - **In Progress Story: User Creation**
    Creating the initial user, John Production Engineer.
  - **In Progress Story: More User Creation**
    Creating two additional users, Quality Inspector Mike and Plant Manager Albert.
- **Data Security - Field Level Access**
  - **In Progress Story: Field Level Access**
    Configuring granular field-level security for sensitive fields such as Payment_Status__c, Discount_Total__c, and Total_Amount__c on AgriEdge_Order__c.

- **Apex Class, Triggers & Testing**
    - **In Progress Story: Create Apex**
      Development of multiple Apex classes and triggers for automation and business logic: OrderTaskCreator, OrderStatusUpdater, OrderTotalUpdater, OrderEmailSender, AgriEdgeOrderShipmentHelper, AgriEdgeOrderTriggerHelper, OrderItemTrigger, OrderPaymentStatusTrigger, AgriEdgeOrderTrigger.
    - **In Progress Story: Create Test Class for All Apex Classes in One Class**
      Development and execution of the comprehensive AgriEdgeOrderTests class to ensure code coverage and functionality of all custom Apex components.
- **Automation**
    - **In Progress Story: Process Builder Creation**
      Creation and activation of the Order Task Creator Process Builder to invoke the OrderTaskCreator Apex class upon new order creation.

6. System Requirements

To ensure optimal performance and access to the Salesforce-driven OMS, the following system specifications are recommended:

6.1. Supported Browsers

- Google Chrome (Latest stable version, recommended)
- Mozilla Firefox (Latest stable version, recommended)
- Microsoft Edge (Latest stable version, recommended)
- Safari (Latest stable version, Mac only)
- Internet Explorer 11 (Limited support, not recommended)
- *Note: At least 2 browsers should be installed on the system.*

6.2. Operating System Compatibility

- Windows 10/11
- macOS (Latest versions)
- Linux (Limited support, browser-dependent)
- ChromeOS (Browser-based usage)

6.3. Hardware Requirements

- **Processor:** Intel Core i3 or higher (or equivalent)
- **RAM:** Minimum 4GB RAM (8GB or more recommended for better performance)
- **Storage:** At least 10GB free disk space
- **Display Resolution:** Minimum 1366 x 768 (1920 x 1080 recommended)

6.4. Network Requirements

- **Stable internet connection:** Broadband, minimum 3 Mbps recommended.
- **No VPN restrictions:** Ensure VPN does not block Salesforce access.
- **Allow Salesforce domains in firewall settings:** (e.g., .salesforce.com, force.com).
