Project Document

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

# Project Overview:

The AgriEdge Order Management System is a customized Salesforce CRM solution designed to streamline and automate the agricultural supply chain process for Or-Mange Ltd. The primary focus of this system is to manage customer orders, shipment tracking, inventory control, and task automation efficiently within a single platform. By leveraging Salesforce's automation capabilities, the solution minimizes manual effort, enhances real-time tracking, and improves communication between customers, suppliers, and delivery teams. Key features include order lifecycle tracking, automatic task creation, inventory availability management, shipment generation, and real-time order status updates.

# Objectives

The main goal of this project is to **streamline the order management process** for agricultural supply chains using Salesforce CRM. By automating key operations such as task creation, payment tracking, shipment handling, and inventory updates, the project aims to reduce manual errors and improve efficiency.

Key objectives include:

1. Automate task assignment based on order status.
2. Track payment and shipment status in real-time.
3. Maintain accurate inventory record
4. Enable timely communication through email alerts.
5. Provide a scalable and maintainable CRM solution for future business growth.

## Phase 1: Requirement Analysis & Planning

Understanding Business Requirements:

The AgriEdge project was developed to address the need for an efficient and automated order management system in the agriculture sector. Business users required a centralized solution to handle customer orders, inventory updates, shipment tracking, and task assignment within Salesforce CRM. The goal was to reduce manual errors, streamline workflows, and ensure timely order fulfilment.

Defining Project Scope and Objectives:

* Build a scalable CRM application to manage the end-to-end lifecycle of AgriEdge Orders.
* Automatically create tasks upon order placement to notify the operations team.
* Update order statuses based on payment confirmation and shipment progress.
* Maintain accurate inventory levels and shipment details linked to each order.
* Ensure the architecture supports future enhancements such as chatbot integration.

Designing Data Model and Security Model

Created the following custom objects:

1. AgriEdge\_Order\_\_c
2. AgriEdge\_OrderItem\_\_c
3. AgriEdge\_Inventory\_\_c
4. AgriEdge\_Shipment\_\_c

 Configured lookup relationships between orders, order items, inventory, and shipments.

 Applied validation rules, field-level security, and sharing settings.

 Set up appropriate profiles, permission sets, and a role hierarchy to manage access control and ensure data confidentiality.

### Phase 2: Salesforce Development – Backend & Configurations

**Setup Environment & DevOps Workflow**

* Configured the Salesforce Developer Edition environment for building and testing the application.
* Followed a modular development approach using Change Sets for deployment.
* Ensured version control of Apex classes and configuration through organized backups and naming conventions.

**Customization of Objects, Fields, Validation Rules, and Automation**

* Created custom fields on all custom objects (AgriEdge\_Order\_\_c, AgriEdge\_OrderItem\_\_c, AgriEdge\_Inventory\_\_c, AgriEdge\_Shipment\_\_c) based on business requirements.
* Implemented validation rules to ensure data integrity, such as:
  + Preventing negative quantities in inventory.
  + Ensuring required fields like Order Status and Payment Status are properly filled.
* Built Process Builder flows and Record-Triggered Flows for automation, including:
  + Auto-task creation when an order is placed.
  + Status updates when payment is marked as “Paid”.
  + Inventory reduction upon order shipment.

**Apex Classes and Triggers**

* Developed reusable Apex classes to handle business logic:
  + OrderTaskCreator: Automatically creates a task when an order is placed.
  + OrderStatusUpdater: Updates order status based on payment confirmation.
  + AgriEdgeOrderShipmentHelper: Manages shipment records and logic.
* Created Apex Triggers:
  + On AgriEdge\_Order\_\_c to update statuses and call Apex helper methods.
  + On AgriEdge\_Shipment\_\_c to maintain shipment linkage and order state.
* Controlled recursion using a helper class AgriEdgeOrderTriggerHelper to avoid infinite loops.

### Phase 3: UI/UX Development & Customization

**Lightning App Setup through App Manager**

* Created a custom Lightning App named **AgriEdge Order Manager** using the App Manager.
* Added all relevant custom objects (Orders, Order Items, Inventory, Shipments) to the app for quick access.
* Customized app navigation with intuitive tabs and icons for ease of use.

**Page Layouts and Dynamic Forms**

* Configured page layouts for each object to display relevant fields based on user roles.
* Utilized **Dynamic Forms** on record pages to show/hide fields conditionally based on record data (e.g., display shipment fields only when the status is “Shipped”).
* Grouped fields into logical sections for better readability and maintenance.

**User Management**

* Created custom user profiles and roles based on access levels (e.g., Order Manager, Inventory Handler).
* Assigned appropriate page layouts, permissions, and field-level access based on the profile.
* Enabled role hierarchy to support record-level sharing based on business structure.

**Reports and Dashboards**

* Built summary and tabular reports to track:
  + Total orders and revenue by month.
  + Inventory usage and reorder needs.
  + Shipment timelines and delivery statuses.
* Created interactive dashboards with visual charts and graphs to provide a real-time overview of the order management process.

**LWC Development (if applicable)**

* No custom Lightning Web Components (LWC) were developed in this version.
* The application relies on standard Lightning components and configuration.

**Lightning Pages**

* Customized Lightning Record Pages for key objects using Lightning App Builder.
* Added components like Highlights Panel, Tabs, Related Lists, and Path to enhance the user experience.
* Ensured mobile responsiveness and fast-loading UI performance.

### Phase 4: Data Migration, Testing & Security

**Field History Tracking, Duplicate Rules, and Matching Rules**

* Enabled **Field History Tracking** on key objects like AgriEdge\_Order\_\_c and AgriEdge\_Shipment\_\_c to maintain an audit trail of critical field changes (e.g., Payment Status, Order Status).
* Configured **Duplicate Rules** for the Order and Inventory objects to prevent duplicate entries.
* Created **Matching Rules** based on custom logic for detecting duplicate orders (e.g., based on customer and order date).

**Profiles, Roles, and Permission Sets**

* Defined custom **Profiles** such as “Order Manager” and “Inventory Supervisor” with object-level and field-level security.
* Established a **Role Hierarchy** allowing regional managers to access data for their respective teams.
* Configured **Permission Sets** to provide additional access (e.g., to Reports and Dashboards) without modifying base profiles.

**Sharing Rules**

* Implemented **OWD (Organization-Wide Defaults)** as private for sensitive objects like AgriEdge\_Order\_\_c.
* Created **Sharing Rules** to extend access to users in the same region or team, ensuring data visibility aligns with business needs.

**Test Classes and Coverage**

* Created test classes for each Apex class and trigger, covering various scenarios:
  + Order creation and automatic Task creation.
  + Updating payment status and triggering shipment logic.
  + Handling shipment deletion and order item adjustments.
* Achieved over **75% code coverage** across all Apex components.

**Test Cases with Input/Output Screenshots**

* Prepared individual test cases for major features:
  + **Order Creation Flow:** Input form and confirmation screen.
  + **Approval Process Execution:** Initial request and approval log.
  + **Automatic Task Creation:** Order submission and related task screenshot.
  + **Trigger Execution:** Before/after screenshots showing data updates.
* Each test case includes:
  + Description
  + Input data
  + Expected outcome
  + Actual result
  + Screenshots

### Phase 5: Deployment, Documentation & Maintenance

**Deployment Strategy**

* **Change Sets** were used to deploy metadata from the Developer Sandbox to Production.
* Components deployed included:
  + Custom Objects: AgriEdge\_Order\_\_c, AgriEdge\_OrderItem\_\_c, AgriEdge\_Inventory\_\_c, and AgriEdge\_Shipment\_\_c
  + Apex Classes and Triggers: OrderTaskCreator, AgriEdgeOrderShipmentHelper, AgriEdgeOrderTriggerHelper, AgriEdgeOrderTrigger, and OrderStatusUpdater
  + Automation: Process Builder for automatic task creation
  + Validation Rules, Flows, and Page Layouts
* Full test coverage was ensured with more than **75% code coverage**, and **7/7 test cases passed** before deployment.

**System Maintenance and Monitoring**

* Regular checks performed for:
  + **Automation errors** using Debug Logs and Flow Error Emails.
  + **Inventory and Shipment records** to ensure business logic runs smoothly post-deployment.
  + **User activity and access issues** using the Setup Audit Trail.
* Admin roles were assigned to manage object-level access, user permissions, and day-to-day operations.

**Troubleshooting Approach**

* Debug Logs and Developer Console were used to identify and fix issues in:
  + Apex Triggers not firing correctly
  + Flow misconfigurations
  + Validation Rule conditions
* Recursion issues in triggers were handled through the AgriEdgeOrderTriggerHelper class.
* All identified issues were documented and resolved step-by-step, maintaining a log for future reference and scalability.

## Conclusion:

The **AgriEdge Order Management Project** provided an effective solution to the challenges of managing agricultural orders and shipments. Using the Salesforce CRM platform, we streamlined the complete order process — starting from order creation and task generation, all the way to inventory adjustments and shipment tracking.

By automating routine tasks with Process Builder and Apex Triggers, the system improved efficiency and reduced manual errors. We ensured data accuracy through custom objects, validation rules, and a strong security framework that aligns with the business requirements.

Custom Apex logic added flexibility and control, especially in handling critical actions like updating order statuses when payments are made or removing shipments when orders are cancelled. With over 75% test coverage and all test cases passing successfully, the solution is stable, reliable, and deployment-ready.

Overall, this project highlights how a well-balanced use of Salesforce's point-and-click tools and Apex development can result in a scalable, maintainable system. It also opens the door for future upgrades like AI-driven insights or chatbot integration to further enhance user experience.