1. Table of Contents
2. Introduction
3. Entity Relationship Diagram (ERD) Overview
4. ERD Considerations
5. Assumptions
6. Salesforce Security
   * + 1. Roles
       2. Profiles
       3. Public Groups
       4. Permission Sets
       5. Organizational-Wide Sharing Defaults
       6. Sharing Rules
       7. User Authentication
7. Object Overview
8. Assumptions
   * 1. Internal Users
9. salesforce.com Components
10. Record Types
11. Page Layouts
12. List Views
13. Workflows
14. Triggers
15. CRUD Rules
16. Salesforce Release Management and Deployment Strategy
17. Overall Landscape
18. Deployment Strategy

# Cleaning Company Project

Create a system for “Cleaning Company” managing a professional cleaning services company that operates on a major metropolitan area.

# Project Plan

## The project will follow a 2 week accelerated development cycle using salesforce.com technology. The table below lists an estimated timeline for the project and key touch points with the client. The client is expected to review and approve the below time line.

## 

| # | Activity | Due Date | Status |
| --- | --- | --- | --- |
| 1 | Requirement Review and Signoff | 5/25/2016 | Send to Client on 5/24 |
| 2 | Initial Design Review and Signoff | 5/27/2016 |  |
| 3 | Client access to Sandbox | 5/30/2016 |  |
| 4 | Review of Project build | 6/3/2016 |  |
| 5 | Signoff on the Sandbox | 6/3/2016 |  |

# Requirement Review

# Based on the initial understanding of the requirements I have come up a number of questions and assumptions. Please update this document with your answers. This is important for me to proceed with the design of the system.

| # | Requirement | Questions/Assumptions |
| --- | --- | --- |
| 1 | Create a system for managing a professional cleaning services company that operates on a major metropolitan area | Assumption:  Cleaning Company intends to use salesforce.com as their Information system and has sufficient license for users.  Assumption:  Cleaning Company DC Metro Area will be used for the initial pilot of this system. |
| 2 | The job delivery unit is always cleaners per day. (e.g.: a private home owner hires one cleaner for one day, a large office hires three cleaners for one day) | Assumption:  There is no dependency on the equipment/cleaning supplies. The only factor that will be tracked in the system will be cleaners.  Question:  Is the number of cleaners per job fixed? Is there possibility that when customer orders a new job they ask for additional cleaners? |
| 3 | The jobs can be booked individually (ad-hoc) or with the following recurrences: weekly, bi-weekly or monthly | Question:  Is there any limit to the number of jobs that can be handled in a day?  How many employees does Cleaning Company have?  Can I assume 1 employee – 1 job/day, and a total of 12 employees |
| 4 | Payment method is always Credit Card. | Assumption:  For the initial version we are not doing any integration with Credit Card authorization service. |
| 5 | The standard rate is 180USD/cleaner/day, recurring jobs have a 10% discount | Assumption:  The orders are taken by a representative and data entered in the system. The representative will verify phone number in the account of the customer will be used to determine if the caller is a repeat customer. |
| 6 | The company employees works on commission, the gross revenue for each job is split 50/50 (50% is company revenue and 50% is the employee pay) The company needs to keep track of: 1. all booked and executed jobs 2 the total revenue generated and the number of jobs executed per client 3. the amount due and paid and the number of jobs executed by each cleaner | Assumption:  The system is only tracking the revenue and cleaner payment. No other expenses ( materials, office expenses, transportation expense etc) are tracked in the system. |
| 7 | Bonus: 1. A dashboard displaying the number of jobs executed and booked for the current week. 2. Display the client address (where the job will be executed on Google Maps) | Assumption: Initial version of the system will not be considered as mobile ready. Subsequent phases will address the mobility aspect of the application. |
| 8 | Extra Bonus: Consider the cleaners are paid every 1st and 15th of each month and design a system that accrues all jobs executed by each cleaner and keep track of all payments made to the cleaner. 1. All jobs executed between the 1st and the 14th of each month is paid on the 15th 2. All jobs executed between the 15th and last day of the month are paid on the 1st | Assumption:  The payment by the client will be automatically applied after the cleaning job is done. Initial release of the system will not be able to validate refunds to customer.  Assumption:  The Payment to the employees will be processed by the back-end accounting system and the initial version of the system does not include integration with the accounting system. |

Deign Options

Option 1 – User of Work Order Standard Object from Spring 16

Salesforce Work Order Standard Objects can be used for field services – but this is more beneficial if the field reps also have salesforce license. Since the license is limited we cannot go this route

* Use Account to capture the Customers
* Use Work Order object to capture Work Orders
* Use of Work Order Line items capture the work order details
  + use 2 types of products – Corporate and Home Cleaning with default units set as 3 workers and 1 worker.
  + Use quantity scheduling and revenue scheduling feature of the Products
* Each field agents be assigned to the work order to complete the work order assignment and send notifications on assignment out of the box.

Option 2

* Use Account to capture the Customers
* Create Contact of type Workers to capture the workers ( create a record type 'Worker' for this )
  + order line items will be a related list for each worker – include some order header data
  + if work is completed ( order is activated )
  + Add payment made/not made to the order line items.
  + Create a make payment button that opens the due payment and the jobs in a related list that can be closed. Based on the date – it opens the related list..
* Use Order object to capture Orders
* Use of Order Line items capture the number of works
  + use products as people ? - cannot be done as workers has to be an entity
  + create monthly, biweekly and weekly version of the 2 products, Corporate and Domestic cleaning.
  + Use quantity scheduling and revenue scheduling feature of the Products
  + use the pricebook feature for discounts – 2 price books / regular and discounts
  + add validation rule that the discounted pricebook can be used only on weekly/monthly orders
* create a relationship from order line item to Workers to capture who did the work assignment before an order can be activated, the worker has to be assigned

Enable Orders = true

Enable Reduction orders = false

Enable Negative Orders = false

Harry.Clean - CEO

Tom.Clean - CFO

Mike.Clean – Scheduler

Name of the Cleaner Workers

Abby, Gail, Molly, Mary, Garret, Alex, Abigel, Martha, Nikki,

|  |  |  |
| --- | --- | --- |
| * [**Glossary**](https://help.salesforce.com/apex/HTViewHelpDoc?id=glossary.htm&language=en_US) * [**Downloadable User Guides**](https://help.salesforce.com/apex/HTViewHelpDoc?id=getstart_help.htm&language=en_US)   Sample Flow That Loops Through a Collection | Salesforce **Sample Flow That Loops Through a Collection** Transfer ownership of accounts from one user to another by using sObject variable collections and loops. The flow already has the required user IDs.   |  | | --- | | Available in: both Salesforce Classic and Lightning Experience | | Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions |   First, create an Account-based sObject collection variable called collAcctJSmith and populate it with all account records that John Smith owns.  Then create a loop that iterates through the collection. For each item in the collection, the loop does the following:   1. Assigns the collection item to the loop variable. 2. Evaluates whether the account has more than 10,000 employees. 3. If the account has more than 10,000 employees, assigns Madison’s user ID to the OwnerId field in the loop variable. 4. If the account doesn’t have more than 10,000 employees, assigns Amber’s user ID to the OwnerId field in the loop variable. 5. Adds the loop variable’s values as a new item in a second collection called collReassignedAccts.   Finally, create a Fast Update element to update the accounts in collReassignedAccts with the new OwnerId after the loop finishes iterating through the collection. |

