Construction Junction

Inventory Management

Software Architecture Document

Version 2.0

SUMMA Technologies

October 1st, 2009
Summa Technologies, Inc.
925 Liberty Ave.
Pittsburgh, PA 15222
(412) 258-3300

Table of Contents

Revision History

Date	Version	Description	Author
October 1st, 2009	1.0	Initial draft	Adriano Wisnik
March 7, 2011	2.0	Final draft	Mindy Schwartz

Project Team

Role	Name	Department
Business Owner	Mike Gable	Executive Director
Business Unit Representatives	Mindy Schwartz	Systems & Development Director
Architect	Adriano Wisnik	Summa

1. Introduction

1.1. Purpose

The purpose of this document is to provide a conceptual overview of the system architecture based on the current state of the Construction Junction Inventory Management requirements.

This document will be used to build a more detailed design of the system architecture as well as identify areas where further research may be necessary to define the system architecture.

1.2. Scope

This document's scope is restricted specifically to a conceptual discussion of the needed software architecture and infrastructure components necessary to implement the Inventory Management application.

1.3. Audience

This document is intended for the project and technical managers, team leads, quality assurance, individual developers, as well as the system architect.

1.4. Definitions, Acronyms, and Abbreviations

- **CSS** – Cascading Style Sheets – Used to describe the presentation of a document written in a markup language like HTML. It separates the content from the presentation.

- **Layer** – A collection of objects that fulfill a specific application concern. For instance, the data access layer is responsible for loading and saving data, the business rules layer is responsible for enforcing business rules.

- **Tier** – A logical or physical grouping of multiple layers within the system. Typical tiers in a multi-tier system are presentation, business (or application), and persistence tiers. The individual tiers may or may not be located on separate machines.

- **CRM -** Customer Relationship Management

- **Apex** - Strongly-typed, object-oriented programming language that allows developers to execute flow and transaction control statements on the Force.com platform server in conjunction with calls to the Force.com API.

- **Visualforce** - A simple, tag-based markup language that allows developers to easily define custom pages and components for apps built on the platform. The components can either be controlled by the same logic that is used in standard Salesforce.com pages, or developers can associate their own logic with a controller written in Apex.

- **Visualforce Controller** - An Apex class that provides a Visualforce page with the data and business logic it needs to run.

- **AppExchange** - A sharing interface from salesforce.com that allows you to browse and share apps and services for the Force.com platform.

1.5. References

The following is a list of resources which provide background on the documented system architecture.

1.5.1. Construction Junction References

- Construction Junction's Toolkit Inventory Screens spreadsheet

- Construction Junction's Project Goals document

- Construction Junction's Inception Findings and Recommendations document

- Construction Junction's Inventory Management System Software Requirements Specification

1.5.2. External References

Although not all may be explicitly referenced, knowledge gained from these resources influenced the architecture of this system.

- Visualforce developer's guide:
  http://www.salesforce.com/us/developer/docs/pages/index.htm

- Apex Code developer's guide:
  http://www.salesforce.com/us/developer/docs/apexcode/index.htm

- Developerforce documentation wiki:
  http://wiki.developerforce.com/index.php/Documentation

1.6. Overview

The remainder of this document is broken down into several key sections. Section 3 provides an overview of the various tiers and layers within the system including both the logical and physical partitioning of the system. Section 4 covers the proposed security model.

Sections 5 and 6 outline a series of common components and application level services which will be provided. This is the first step in the system's functional decomposition process. Each of the common components is briefly discussed at a conceptual level.

Finally, Section 7 includes a list of the third-party tools and components recommended for use in the project along with their approved versions.

The Appendix contains diagrams depicting the overall architecture and main flows of the application, as well as its hardware requirements.

2. Architectural Forces

The inventory management application will consist of a database, multiple user interfaces (such as desktop application, mobile interface for Pickup/Decon crews, cash register), receipt and item label printers, as well as integration points with QuickBook POS and the Construction Junction website.

During the Inception Phase of the project it was determined that Construction Junction would develop a custom inventory management application building on the SalesForce.com platform that it already had access to as a result of its ongoing Customer Relationship Management project. Built into SalesForce.com is an application development platform called Force.com. Force.com allows custom applications to be built in the same environment as the Customer Relationship Management application that Construction Junction is already using.

Developing an inventory management application using Force.com gives Construction Junction the ability to implement the custom features needed, without the need to invest in additional hardware or software licenses. It also has built in support for mobile applications and simplifies integration with external applications such as QuickBooks POS and the Construction Junction website.

Force.com also makes sharing the software to other similar businesses using Salesforce.com very easy through the AppExchange, which allows custom applications to be distributed to other companies, either free or for a fee.

Maintaining the customer contact information and inventory management in a single location eliminates an integration point and simplifies implementation. Force.com also has a number of features, such a support of mobile devices and touch screen interfaces built in which will also make development easier and faster.

This approach is also aligned with Construction Junction's desire to reuse the application and market it to other similar companies.

3. System Layers

The Inventory Management application will use a multi-tier and multi-layer architecture. The following diagram illustrates how the various tiers are partitioned within the system:
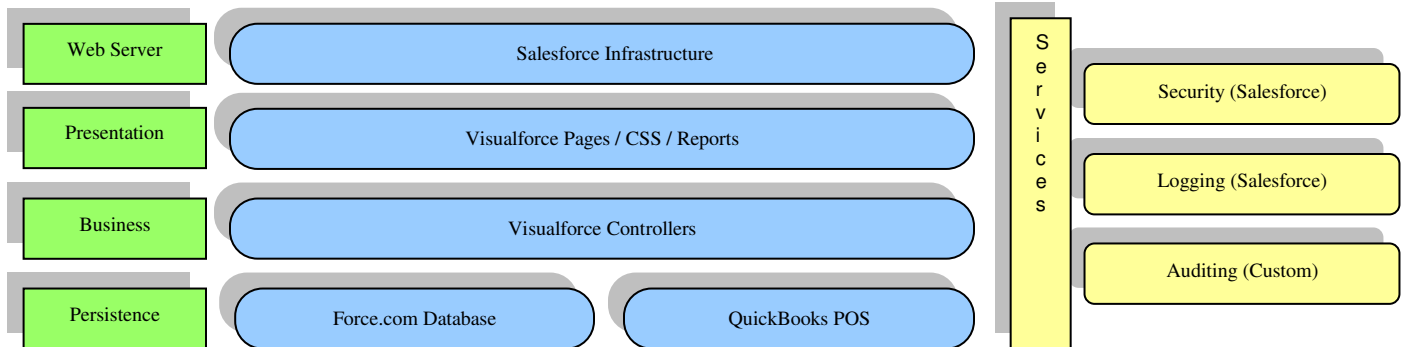

Figure 1 – System layers and partitioning

The three tiers within this model are:

- **Presentation Tier** – Handles the display and collection of information from the user.

- **Application Tier** – Applies business rules and data transformations.

- **Persistence Tier** – Stores and retrieves the data from the data store (e.g. relational database) and handles communication between system components.

Each tier is separated into one or more layers, with each layer responsible for a specific set of concerns. In addition to the various tiers, there are a series of cross-cutting concerns (software functions which exist in each layer). These cross-cutting concerns are discussed in sections 6, 7, and 8.

3.1. Presentation Tier

The presentation tier is based on the Visualforce implementation of the Model View Controller (MVC) pattern. It is responsible for displaying information to the user and receiving input from the user. The presentation tier can be broken down into the following layers:
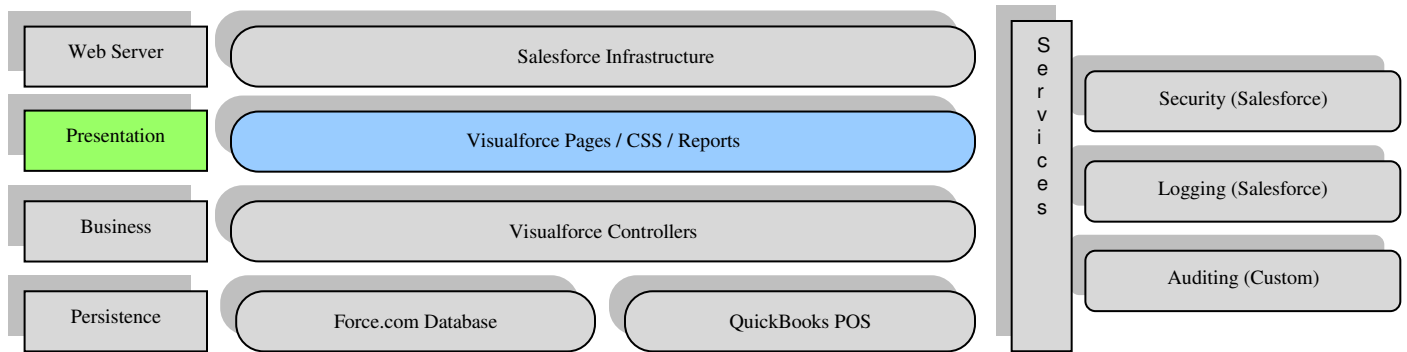
Figure 2 – Presentation Tier

- **Visualforce pages** – A developer creates Visualforce pages by composing components, HTML and optional styling elements on the Force.com platform. Each page is then accessible by a unique URL.

- **Cascading Style Sheets (CSS)** – CSS provides web applications with a mechanism to separate information from the display rules. Effective use of CSS will enable a skinnable look-and-feel.

- **Reports** – Salesforce-generated reports and dashboards.

3.2. Application Tier

The application tier is responsible for performing server-side validation, applying business rules, and transforming objects from the persistence tier into objects usable for the presentation tier. This tier is broken down into the following key layers:
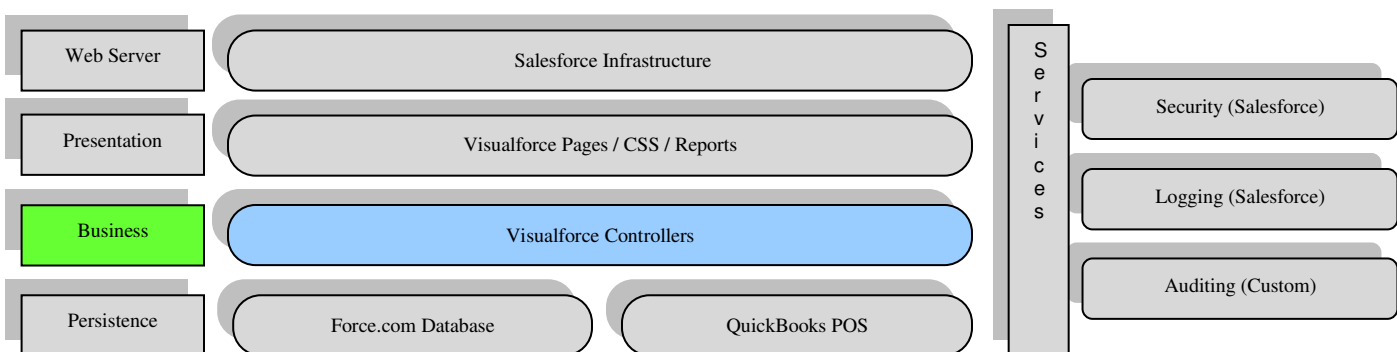

Figure 3 - Application Tier

- **Visualforce controllers** – Custom controllers, written in Apex, deliver the flexibility to define your own logic, navigation, algorithms, and database and web services interactivity. With the stateful programming model provided by Visualforce, custom controllers can maintain state between pages making development of wizards straightforward.

3.3. *Persistence Tier*

The persistence tier is responsible for the saving and loading of data. The persistence tier is made up of the following layers:
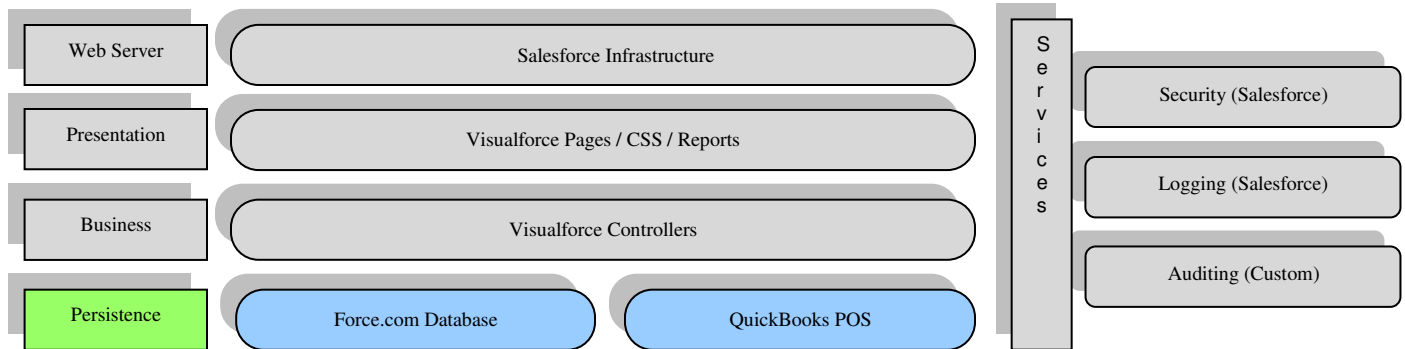

Figure 4 – Persistence Tier

- **Force.com database -** The Force.com Database uses objects to store data. Objects contain the functionality you expect in a table and more, but you should be aware that the difference also points to a difference in functionality. The object comprises a number of fields. Objects can be related to other objects, with relationship fields mapping records in one object to records in another.

- **QuickBooks POS** – Inventory information is synchronized between the Force.com database and the QuickBooks POS database. Inventory items received into the inventory and changes to existing inventory items are also applied to the QuickBooks POS database. And changes to the QuickBooks POS database made as a result of item sales are also applied to the Force.com database.

4. Security

The Inventory Management application requires a stringent yet flexible security model. Salesforce security and Force.com's role and profile-based access control provides that required security model.

4.1. Authentication

All users must log in to the application through Salesforce. When they request a URL from the application, Salesforce's security framework intercepts the request and redirects the user to the Salesforce login page. Salesforce verifies the identity of the user and then redirects the user to the originally requested page. The user is then able to access the application.

4.2. Authorization

All users in the system are provided with a profile and role that allows access to the Inventory Management system. From that point, all user authorization is managed by a combination of profile and role access control rules defined with Salesforce, and fine-grained access control logic provided by the application. User Identity information is retrieved from Salesforce using Force.com's standard Apex API as documented in the Force.com Apex Code Developer's Guide.

The following sections describe the various components of the security model.

4.2.1. Security Model

The Force Platform provides access control based on user profiles and roles.

Profiles control a user's object- and field-level access permissions. A user can't be defined without being assigned to a particular profile, since the profiles specifies the apps and tabs that appear when he or she logs in, among others.

Roles, on the other hand, primarily control a user's record-level access permissions through role hierarchy and sharing rules. Although a role assignment isn't required when a user is defined it does simplify the definition of record-level permissions.

Because profiles control object- and field-level access whereas roles influence record-level access, a user is typically assigned to one of each.

4.3. Auditing

All transactions will need to support some form of auditing. Depending on the type of request and the system processing involved one of the following tools may be used to implement the application's auditing policy:

- **Database Auditing** – Data updates will automatically add audit information to each record for any insert or update to the database. This will include the user creating or modifying the record and the creation and modification date. This provides auditing only if the persistence layer is accessed.

- **Application Based Auditing** – Auditing can be performed during the invocation of certain sensitive functions that require auditing information to be recorded. Examples of such operations are changing an inventory item's price or manually changing an item's on-hand quantity.

- **Debug/System Logging** –Salesforce.com logging framework automatically records system resource information as well as application debugging information. The System

Log may also contain information about workflow rules, validation rules, assignment rules, escalation rules, and approval processes.

It is important to note that non-technical users will need access to the auditing information so developers should take steps to ensure that the information contained in the audit logs is human readable or system parseable.

5. Common Components

The architecture defines a number of common components which can be used by the entire Inventory Management application. This section will be added to as more requirements are elicited. These services include:

- Email Delivery
- Common Controls
    - Pop-up Calendar
    - Table with Paging, Sorting and Controls
    - Selection Matrix
    - Barcode
- Common Validators
    - Email
    - Phone
    - Zip
    - Date
- Logging
- Frameworks
    - Navigational Framework

5.1. Email Delivery

Email services are provided by the Salesforce platform. Emails are sent using Visualforce by creating a custom controller which uses the ApexMessaging.SingleEmailMessage class, which handles the outbound email functionality available to Salesforce.com.

5.2. Common Controls

Various user interface controls will be reused throughout the application.

5.2.1. Pop-up Calendar

A pop-up calendar component will be used to input dates in the system. The pop-up displays a calendar that the user can select a date from. Additionally, the user can enter the date by hand and the date will be validated by the application.

5.2.2. Table

Multiple parts of the application will need to support tabular data. The table control allows sorting the data in the table by column. Sorting of data is across the full data set, not what is displayed in the User Interface alone. The table control also supports paging of data for large data sets.

5.2.3. Selection Matrix

Multiple parts of the user interface require a user to make a selection of an inventory category by navigating the inventory using a matrix view of departments, categories and sub-categories. A reusable interface component should be provided for such a data structure, with a data model object backing it.

5.2.4. Barcode

Barcode generation is required in various areas of the application, and should therefore be available as a reusable component. Examples of barcode uses are: item labels, acquisition labels, acquisition slips and member cards.

5.3. Validators

To support validation of user entered data or consumed data from other systems, various validators will be used in the system. Examples are:

- Email

- Phone

- Zip Code

- Date

5.4. Logging

Effective usage of logging enables faster resolution of customer issues, defect analysis, and monitoring of system stability. The application will use the built-in Force Platform logging capabilities by invoking the system APIs for debugging. The appropriate logging level should be used in each case: ERROR, WARN, INFO, DEBUG, FINE, FINER or FINEST.

.

5.5. Frameworks

5.5.1. Navigational Framework

Visualforce's model based on custom pages and controllers will be used to apply a consistent navigational framework throughout the application.

6. General Architectural Guidelines

6.1. External System Interfaces

The Inventory Management application relies on external systems of record for specific data elements. This integration to these other systems should occur in as transparent a way as possible.

The interfaces between external systems should be designed in such a way as to minimize the impact of downtime on the Inventory Management application. The degree and feasibility for which this is possible will have to be assessed on a case-by-case basis as part of the Construction phase.

6.1.1. Integration with QuickBooks POS

Data will be copied bi-directionally between the application in Salesforce.com and QuickBooks POS on a scheduled basis according to the following rules:

- New **Inventory** records in Salesforce will be copied to QuickBooks POS
- Updated **Inventory** records in Salesforce will be updated in QuickBooks POS
- New **Inventory** records in QuickBooks POS will be copied to the inventory database in Salesforce
- Updated **Inventory** records (such as sales) in QuickBooks POS will be updated in the inventory database in Salesforce

The batch programs that execute the synchronization of inventory data between QuickBooks POS and the inventory application in Salesforce should be implemented to allow for Construction Junction to maintain its typical business flow where items are at times, sold immediately following receiving them.

Besides being executed on a scheduled basis, the batch programs that perform the data synchronization should also be able to be invoked on demand at any time by the Construction Junction staff.

6.2. Exception Handling

Proper exception handling is essential for large-scale enterprise class applications. The Inventory Management application will provide appropriate exception handling policies. The system architecture detailed design and the development guidelines will describe the exception handling policy in greater detail.

The exception handling framework must provide for both checked and unchecked exceptions. When developers define custom exceptions they should apply the following rules to determine if an exception is a checked or an unchecked exception:

- If the developer must take a specific action once the exception occurs it should be a checked exception.

- If the developer has insufficient information on how a problem should be correct or there is no automated error recovery available (besides rolling back a transaction and notifying the user) then it should be an unchecked exception.

- Security exceptions should be unchecked.

Exceptions should truly be exceptional. For instance, if the "exceptional" state is expected to occur on a regular basis, it is not an exception, instead it is a possible outcome.

Custom exception classes written for the Inventory Management application will extend one of two classes:

- A base class for application specific, checked exceptions.

- A base class for application specific, unchecked exceptions.

6.3. *Data Validation*

Data validation can be broken down into three categories:

- **Attribute Level** – These are rules for domain checking (e.g. minimum and maximum value, alpha-numeric, e-mail address, etc.).

- **Object Level** – Rules that define how fields within the same object are validated (e.g. arrival date must be earlier than departure date).

- **Graph Level** –Rules that apply across an entire object graph (e.g. an Acquisition must have a donor, date and item set defined).

The Visualforce controller layer will be responsible for performing all three levels of validation. By ensuring that all requests eventually route through the controller layer and ensuring that it enforces all three levels of validation ensures consistent application of business rules.

7. Third Party Libraries and Components

The following is a list of the various third party components and libraries being used as part of the Inventory Management project. This list also contains the specific versions which will be used for the project.

7.1. Required Tools

Tool	Version	URL	Comments
Force.com Platform	Winter '10	http://developer.force.com/releases/release?key=Winter10	
Visualforce	17.0	http://www.salesforce.com/us/developer/docs/pages/index.htm	
Apex	17.0	http://www.salesforce.com/us/developer/docs/apexcode/index.htm	
Eclipse IDE	3.3.x or later	http://www.eclipse.org/downloads/moreinfo/java.php	See Force.com IDE for supported versions
Force.com IDE	16.0 or later	http://wiki.developerforce.com/index.php/Force.com_IDE_Installation_for_Eclipse_3.3.x	
QuickBooks POS SDK	2.0	http://developer.intuit.com/technical_resources/default.aspx?id=1494	
MS Visual Studio	2008	http://msdn.microsoft.com/en-us/vstudio/default.aspx	For QuickBooks POS integration development

8. Appendices

8.1.   Application Overview

The Inventory Management application will act as a hub, integrating with existing tools that Construction Junction is already using including CRM, QuickBooks POS, constructionjunction.com, eBlast email marketing/VerticalResponse, as well as providing various points of entry within Construction Junction, including office computers, touch screen terminals on the loading dock, and mobile units for pickup and decon.
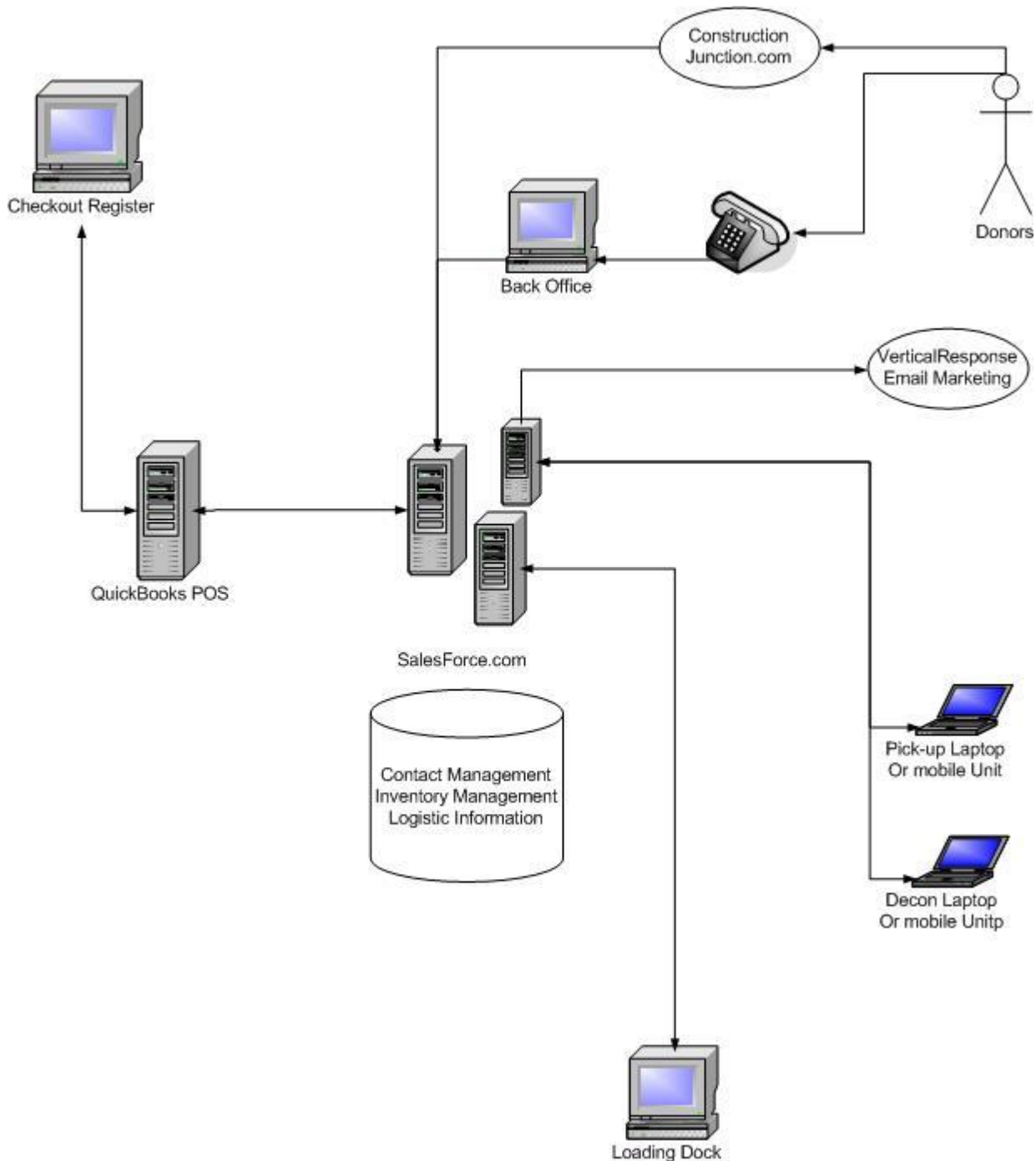

Figure 5 - Application Overview

8.2.   *Workflows*

The figures below detail the workflows for each of the donation methods (drop off, pickup, and decon) and illustrate how inventory will be received and tracked through the application from initial contact through sale of the product.
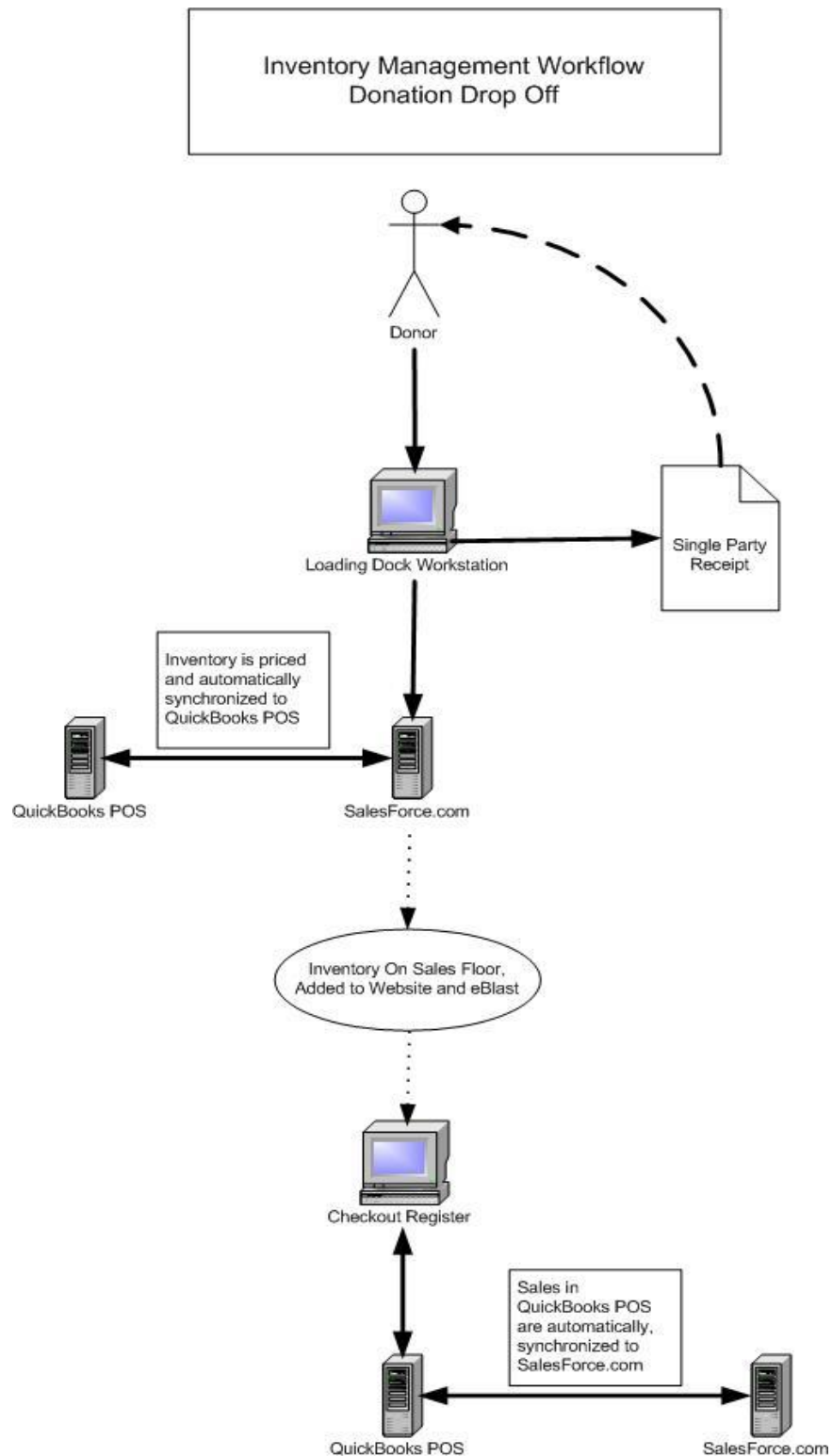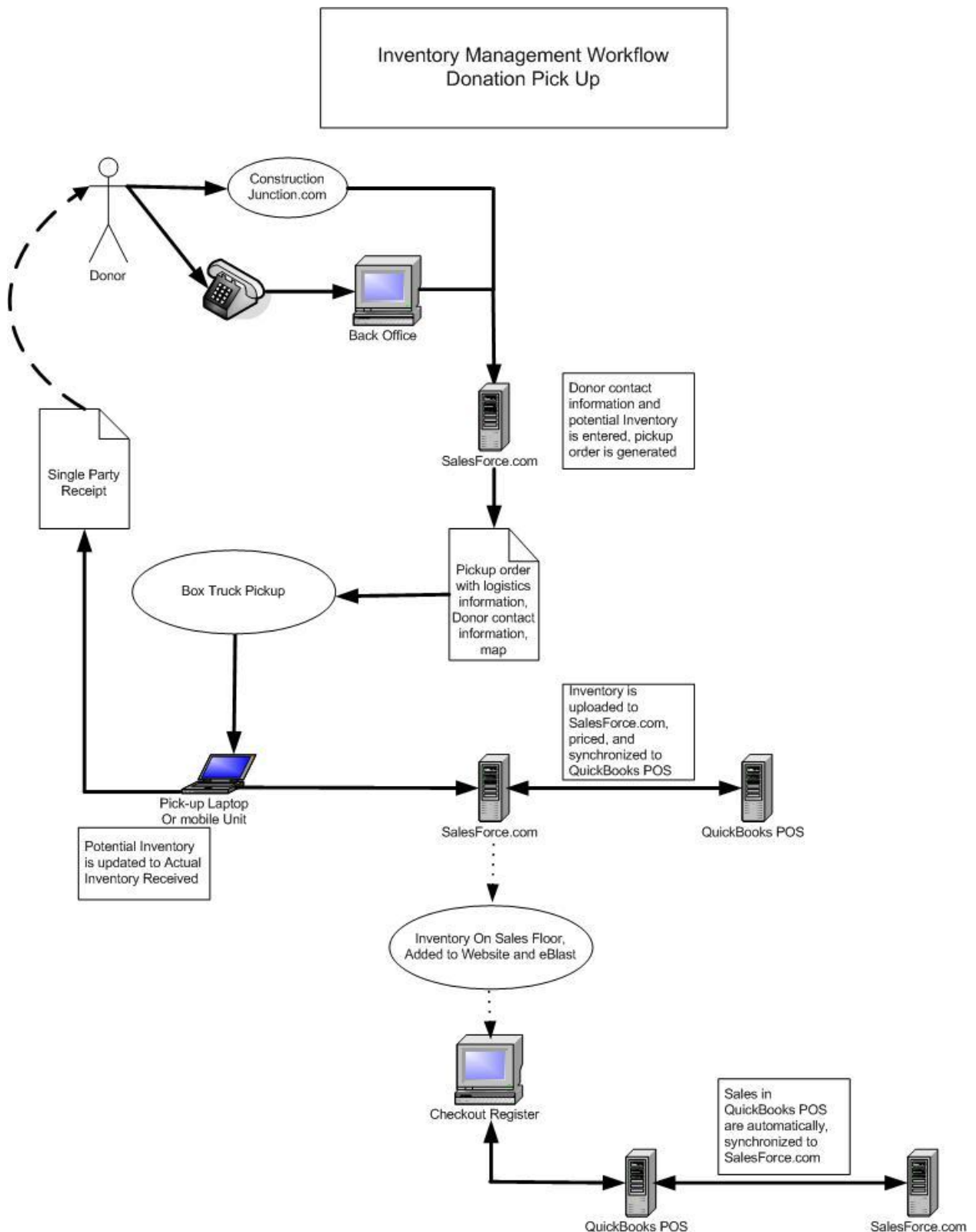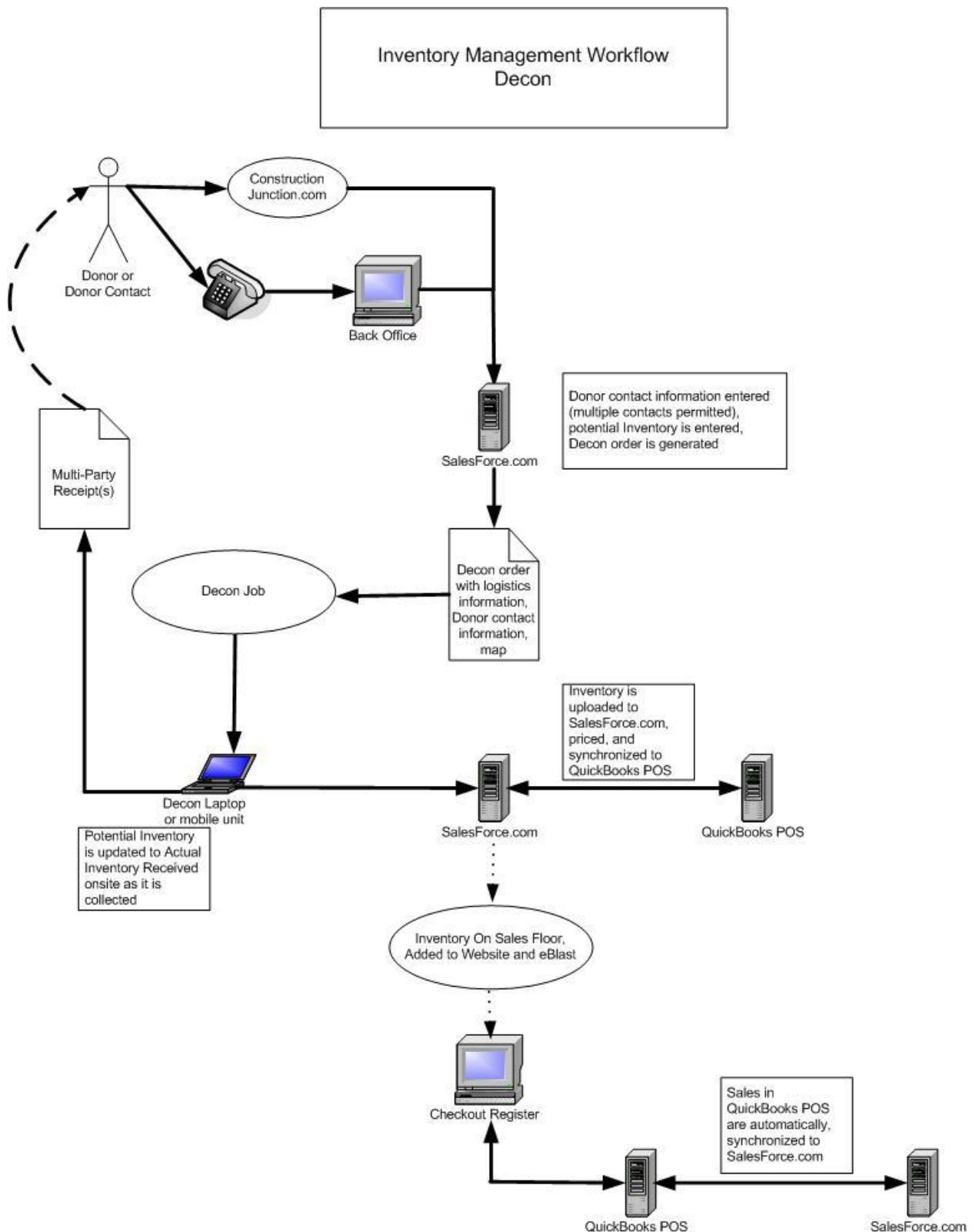

Figure 6 - Drop Off Workflow

Figure 7 – Pickup Workflow

Figure 8 - Decon Workflow

8.3. Hardware Requirements

The following are the hardware components required to implement the documented requirements and support Construction Junction's current operations and near future expansion plans.

Receiving docks – 5 workstations total

- 2 workstations at the dock - for drop-off processing;

  1 regular printer for donation receipts;

  1 label printer (zebra printer);

- 2 workstations in receiving office – 1 for pickup/decon processing and 1 for the receiving manager;

  1 color printer for signage and donation receipt printing;

  1 label printer (zebra printer);

  1 member card printer;

- 1 workstation in warehouse operations office – for drop-off, pickup and decon processing;

  1 regular printer for donation receipts;

  1 label printer (zebra printer);

Customer Service area - 2 workstations total

  2 workstations for various activities: POS, donation scheduling, inventory viewing, etc;

  1 cash drawer;

  1 receipt printer;

  1 regular printer for donation receipt printing;

  1 member card printer;

Checkout Registers (2) – 2 workstations total

  2 workstations for POS;

  2 receipt printers;

  2 cash drawer;

Pickup and Decon Crews (2) – 2 mobile units total

  2 mobile units, 1 for each crew;

  2 regular printers for donation receipts, 1 for each crew;

Pickup and Decon Managers (2) – 2 mobile units total

   2 mobile units, 1 for each manager

Mobile unit for general warehouse management and e-blast scanning – 1 mobile unit total

   1 mobile unit

   1 digital camera – For taking and uploading pictures of items for e-blasting

Notes:

- All workstations are touch-screen workstations;
- All workstations are equipped with bar code readers;
- All mobile units are fully functional laptops.