

**COT 6931 Server Side Design Document**



Food Giant Sales Flyer Generator



Table of Contents

[Revision History 3](#_Toc478898830)

[Section 1 Introduction 3](#_Toc478898831)

[Section 1.1 Project Purpose 3](#_Toc478898832)

[Section 1.2 Server Side Summary 3](#_Toc478898833)

[Section 1.3 Requirements Satisfied 3](#_Toc478898834)

[Section 2 Server Side Design Structure 4](#_Toc478898835)

[Section 2.1 Design Overview 4](#_Toc478898836)

[Section 2.1 Server Side Classes 5](#_Toc478898837)

[Section 2.2 Additional Components 5](#_Toc478898838)

[Section 3 Detailed Design 6](#_Toc478898839)

[Section 3.1 Flyer Creator Program Classes 6](#_Toc478898840)

[Section 3.1.1 Overview of Flyer Creator Program 6](#_Toc478898841)

[Section 3.1.2 View 7](#_Toc478898842)

[Section 3.1.3 ViewModel 7](#_Toc478898843)

[Section 3.2 Flyer History Program Classes 7](#_Toc478898844)

[Section 3.2.1 Overview of Flyer History Program Classes 7](#_Toc478898845)

[Section 3.2.2 View 8](#_Toc478898846)

[Section 3.2.3 ViewModel 8](#_Toc478898847)

[Section 3.3 Basic Flyer Template Program Classes 8](#_Toc478898848)

[Section 3.3.1 Overview of Basic Flyer Template Classes 8](#_Toc478898849)

[Section 3.3.2 View 8](#_Toc478898850)

[Section 3.3.3 ViewModel 8](#_Toc478898851)

[Section 3.4 Database Structure 8](#_Toc478898852)

[Section 3.4.1 Flyer History Table 8](#_Toc478898853)

[Section 3.4.2 Item List Table 10](#_Toc478898854)

[Section 3.5 Database Interface 10](#_Toc478898855)

[Section 3.6 Database Maintainer Program Classes 11](#_Toc478898856)

[Section 3.6.1 Overview of Database Maintainer Program Classes 11](#_Toc478898857)

[Section 3.4.2 View 11](#_Toc478898858)

[Section 3.6.3 ViewModel 11](#_Toc478898859)

[Section 3.7 FlyerDataModel 12](#_Toc478898860)

[Section 3.8 Login Credentials and Program Selector 12](#_Toc478898861)

[Section 4 Terms of Reference 12](#_Toc478898862)

# Revision History

|  |  |  |
| --- | --- | --- |
| Revision # | Date | Changed Items |
| 1 | 02/26/2017 | Initial Document Creation |
| 2 | 03/04/2017 | Added Information on Design Strategies |
| 3 | 03/26/2017 | Had updates between last and now so haven’t updated revision history. Added new sections, reworked every section of document |
| 4 | 04/01/2017 | Added more information to Sections 3.2-3.5 |
| 5 | 04/02/2017 | Added more to Section 3.3. Cleaned up minor grammar issues |
| 6 | 04/15/2017 | Updating Items per client review. Combined Client Side Design Document into this document and refactored some sections due to Virtual Machine change. |

# Section 1 Introduction

This Server Side Design document will contain much of the Food Giant Flyer Generator’s design structure. The Server Side programs are designed to allow a District or Store Manager to handle all actions with previewing, printing and maintaining all data associated with the store flyers. The Server Side programs will connect to and retrieve data from a locally hosted SQL database and convert this data into information about the flyer. This will allow the manager to choose the items they need to populate a flyer with the name, pricing, and images of the selected items. District Managers will have the additional functionality of maintaining the SQL database by adding new Food Giant Inventory items to it and viewing created flyers.

## Section 1.1 Project Purpose

The Food Giant Flyer Creation program is designed to allow Food Giant Store Managers more control over selling their excess inventory. This program will contain an easy to use interface that allows Managers to create custom flyers that contain items and pricing of their choosing, after approved by district managers.

## Section 1.2 Food Giant Flyer Creation Summary

These Food Giant Flyer Creation Programs are separated into different “.cs” (C#) and “.xaml” (XAML) classes to understand their functionality more easily and perform maintenance on these programs. The following list of programs is as follows:

* Flyer Creator
* Flyer History
* Basic Flyer Template
* Database Interface
* Database Maintainer
* Program Selector

Also, this document will cover the tables and fields in the SQL database and explain what each field’s purpose is in the overall program. This document will also explain the design for Database Interface, the C# class that will connect to the SQL database.

## Section 1.3 Requirements Satisfied

The Food Giant Design Document applies to and satisfies the following requirements:

* SR 1.1
* SR 1.1.1
* SR 1.1.2
* SR 1.1.3
* SR 1.2
* SR 1.2.1
* SR 1.2.2
* SR 1.2.3
* SR 1.2.4
* SR 1.3
* SR 1.3.1
* SR 1.3.2
* SR 1.3.3
* SR 1.4
* SR 1.4.1
* SR 1.4.2
* SR 2.1
* SR 2.2
* SR 2.3
* SR 2.4
* SR 2.4.1
* SR 2.4.2
* SR 3.1

# Section Food Giant Flyer Design Structure

## Section 2.1 Design Overview

The overall program will follow a Model View ViewModel (MVVM) design.

All Views will be XAML (Extensible Application Markup Language) classes that are design to contain as little “logic” code as possible and instead only handle showing the visual components to the Manager.

The ViewModels will be “.cs” classes that primary drive each program. The View Models will contain much of the code for this program. The View Model will handle all logic in the code and instantiate the other classes. Any events take by the manager (button click, item select, etc.) will inform the View Model of the action and allow it to handle the logic behind what to do on each of these actions. This MVVM design focuses on View Model first, which means the ViewModel will handle instantiating the View, Models and in some cases, the other View Models.

The Model class will be small “.cs” classes that are be little more than declaring an object and the variables that are required to create this object.

To accomplish this MVVM design effectively, we will be using the Caliburn Micro framework, a completely free Application Program Interface (API) that allows us to bind very quickly and effectively ViewModels, Objects, and Events together with very little redundant code. This will reduce the amount of code that will be required to create this project, and improve maintainability for enforcing coding standards. To use Caliburn, we will enforce class naming for all classes created in the Flyer Generator program. This naming convention is defined as:

1. View – *ClassName*View
2. ViewModel - *ClassName*ViewModel
3. Model - *ClassName*Model

This allows a new team to quickly identify what the class’s overall purpose is just from its name, making maintenance easier.

Variable naming conventions will be enforced and will be defined in a short “Coding Standards” document. This is again to enforce a consistent design and assist with maintainability.

## Section 2.2 Food Giant Flyer Deployment

These section outlines how the following requirements are met:

* SR 1.1
* SR 1.1.1
* SR 1.1.2
* SR 1.1.3

The Food Giant Flyer program will be hosted on Microsoft Azure, a Cloud-Computing platform that can also create Virtual Machines as needed. Food Giant Managers will connect to the application by entering the application’s URL, which will prompt a standard Windows Login window. This way the host machine will need no other requirement than internet access, eliminating the compatibility issues resulting from the unknown software and hardware specifications of the store computers.

The program will exist on each virtual machine and will link to the Azure Active Directory Services to ascertain the Manager’s permission levels. The program will not allow anyone that is not at District Manager level to access the Flyer History or Database Maintainer classes.

On startup, the program will request the user name and account privileges from Azure Active Directory Services. If the Manager has been added to the District Manager group, the program will start with the Program Select Window. If they are not a District Manager, the program will always start with the Flyer Creator program.

Only a system admin or an existing District Manager can add other Managers to the District Manager group. This is to ensure a Store Manager cannot acquire a District Manager’s permissions without their approval. This also allows Store Managers get more permissions if they are promoted, and allows a District Manager to lose their permissions if they leave Food Giant for whatever reason.

## Section 2.1 Food Giant Flyer Classes

The Server Side will consist of a single “.sln” (solution) file, FoodGiantFlyer. The solution file will hold all the individual programs that consist of the entire Food Giant Flyer Generator program. The FoodGiantFlyer solution will consist of the following classes:

Views:

* FlyerCreatorView
* BasicFlyerTemplateView
* DatabaseMaintainerView
* FlyerHistoryView
* ProgramSelectionView

ViewModels:

* FlyerCreatorViewModel
* BasicFlyerTemplateViewModel
* DatabaseMaintainerViewModel
* FlyerHistoryViewModel
* ProgramSelectionViewModel

Models:

* FlyerDataModel

Others:

* AppBootstrapper
* DatabaseInterface

## Section 2.2 Additional Components

In addition to the classes listed in the previous section, the Server Side program will contain the following additional components:

* SQL Database – FoodGiantItemSQLDatabase.mdf. This database will contain all data for the Food Giant Inventory items entered in by an admin and contain the completed flyer history. This is an integral part of the FlyerCreator, DatabaseMaintainer and FlyerHistory programs.
* Images Folder – Located In subfolder under exe folder. This folder contains all added images by an admin. This is an integral part of the FlyerCreator classes.

These components will be discussed further in the detailed design section where applicable to the classes that require their usage.

# Section 3 Detailed Design

This section will detail the purpose of each class, the SRS requirements it affects and a description of the program’s components.

## Section 3.1 Flyer Creator Program Classes

### Section 3.1.1 Overview of Flyer Creator Program

The Flyer Creator program will handle displaying Food Giant Inventory Items to the manager. These Food Giant items will be retrieved from the SQL database using the Database Interface class. The manager will select the flyer template they wish to use, select the flyer sale dates, change the price for each item to what they wish for the sale price to be, then select the images they wish to use for the flyer.

After the manager has selected the items they want to use for the flyer, this program will pass the selected items to the select flyer template page to generate a printable flyer.

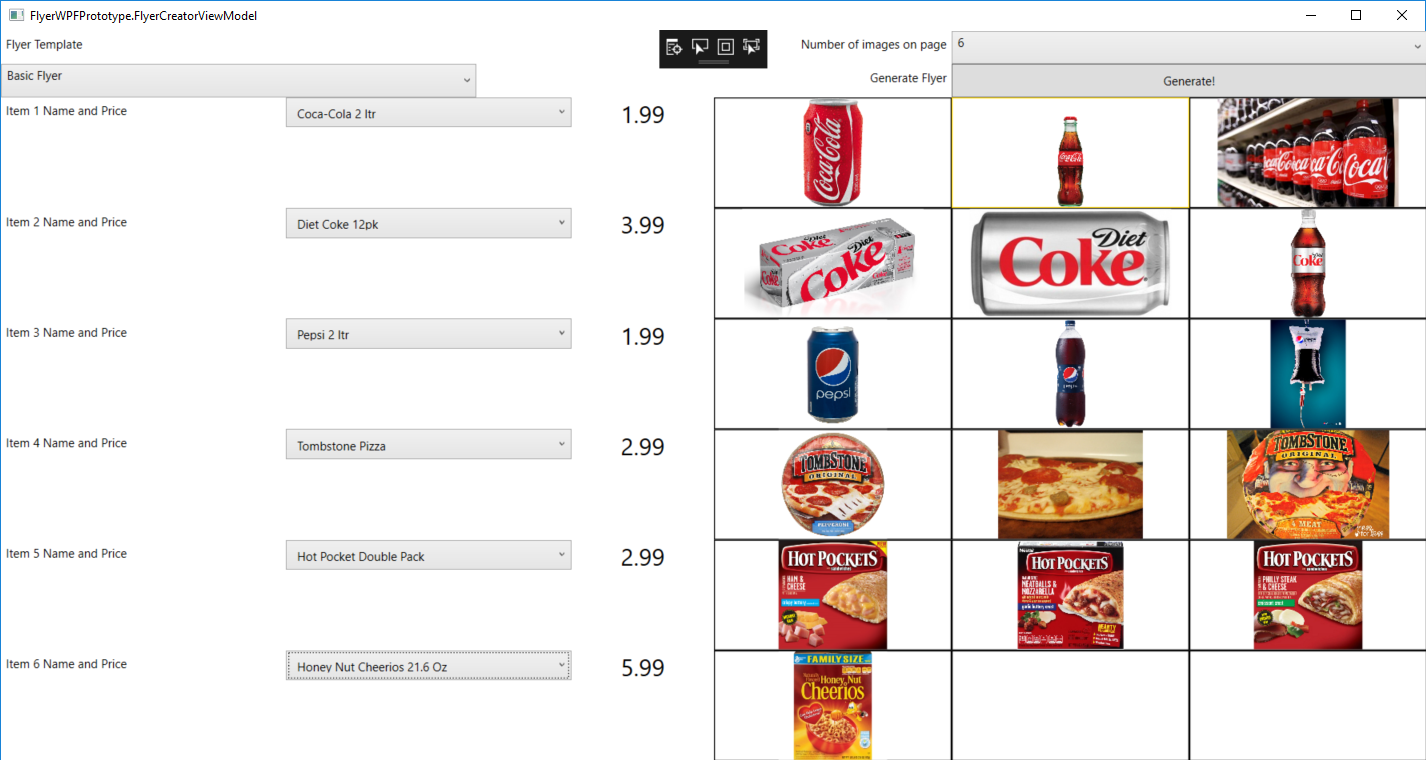


Image 1.1 Flyer Creator Prototype Picture

These classes satisfy requirements

* SR 1.2
* SR 1.2.1
* SR 1.2.2
* SR 1.2.3
* SR 1.2.4

### Section 3.1.2 View

The FlyerCreatorView will contain all the Visible Components that the manager can interact with. All logic handling the manager events will reside in the FlyerCreatorViewModel, keeping in line with the MVVM standard.

The View will contain multiple drop down menus where the manager can select the number of items they wish to display on the flyer, the store type, the store’s address and phone number, the flyer type they want, and the items they wish to use for the flyer.

The Number of Items field will be a drop-down menu containing a sequential list of numbers starting at one and ending at fifteen as per SRS requirement 1.2.4.

The Store Type field will be a drop-down menu containing the following selections:

* Piggly Wiggly
* Food Giant
* Pic-N-Sav
* Citronelle Marketplace

The Store Address and Store Number field will be simple Text Boxes where the manager can enter their store location and phone number. This section may contain functionality in the future for a Software Developer to pre-generate these fields based on a Store Location database.

The Flyer Type field will be a drop-down menu containing only one following selection, Generic Flyer. This option will be expandable to allow a Software Developer to add more flyers as the program matures. The Flyer Creator View Model will contain a list of selectable flyer types.

Each Selected Item will populate a Text Box containing a default price value of “$0.00”, a drop down menu containing the Item Size, a Text Box containing the Item Description (if applicable) and one to three images that the manager can select from to use for the flyer.

After selecting the price option, the manager will have the ability to select an item size via a drop-down menu.

The Item Size drop down will be a pre-generated box containing the following selections:

Item Size box will contain the following options:

* Per Pound
* Each
* Jumbo Pack
* Family Pack

The manager will also be able to add in their own custom text to an Item Description field, which will be a TextBox. The manager will have to make sure they enter in the correct data, since it could be any additional text.

Next, a manager will select the item they want for the flyer by simply clicking the image, which will highlight the border around it gold. If there is only one image, that image will be selected (highlighted gold) by default.

Finally, there will be two checkboxes that contain the following selections:

* While supplies last
* No rainchecks given

These selections will be entered in as Boolean values for all Flyer Templates. If selected, they will add a small image to the next to the flyer address section that will look like this  


### Section 3.1.3 ViewModel

The FlyerCreatorViewModel will be the primary C# class used for the Food Giant Program.

The FlyerCreatorViewModel will contain all the actions and logic for the FlyerCreatorView. It will also instantiate the View and populate all the data for the visible controls, keeping in line with the MVVM standard.

This class will also use the DatabaseInterface class to retrieve data from the SQL database to populate in the drop-down boxes.

This class will contain validation for every user input in order to reduce the chance of an unhandled exception occurring.

When a manager selects the number of images to display on the page, this class will hide the extra selections.

## Section 3.2 Flyer History Program Classes

### Section 3.2.1 Overview of Flyer History Program Classes

The Flyer History program allows a District Manager to quickly examine preexisting flyers created by any Manager. This program will link to the SQL database and retrieve a formatted list of every created flyer that the Manager can then filter by date or manager. This data will be sent to the correct Food Giant template to recreate the flyer without taking up excessive hard drive space by backing up the full copy of each flyer. This allows a District Manager to verify what sale prices were set by the Store Manager for company recordkeeping.

These classes satisfy requirements

* SR 1.3
* SR 1.3.1
* SR 1.3.2
* SR 1.3.3

### Section 3.2.2 View

The Flyer History View will contain all the controls required to view an existing flyer. This program will exist as a Window Control. There will be a Drop-Down box containing a list of all created flyers from the database. There will also be a Calendar and another Drop Down box that filters by date and manager, respectively. There will also be a generate button that will show the flyer data.

### Section 3.2.3 ViewModel

The View Model will contain all the event handlers and validation logic for the program. This class will validate all options selected by the Manager and connect to DBInterface to return data from the database used to populate the controls discussed in the view.

## Section 3.3 Basic Flyer Template Program Classes

### Section 3.3.1 Overview of Basic Flyer Template Classes

The Basic Flyer Template User Control will be a small project that gets data sent to it from the Flyer Creator that is used to create the actual store flyer. This class will only be instantiated from the Flyer Creator class. The Manager will be able to either right click the User Control and print, click a print button, or press Control + P. The print feature will use the standard windows printing functions, exactly like if you print any standard document.

These classes satisfy requirements:

* SR 1.4
* SR 1.4.1

### Section 3.3.2 View

The Basic Flyer View will be based on store flyers I have been sent from my client. Most of the effort for this User Control will exist in the View, since it relies on many different types of flyer images sent by a Manager. The Flyer’s appearance is exactly how the printed flyer will look, so the View must have a significant amount of time spent working on its appearance.

### Section 3.3.3 ViewModel

The View Model will contain all the event handlers and validation logic for the program. This class will validate all items sent to it from the Flyer Creator and bind to each control.

## Section 3.4 Database Structure

The SQL database satisfies requirements:

* SR 2.1
* SR 2.2

The SQL Database will contain the following tables:

* FlyerHistory
* ItemList

### Section 3.4.1 Flyer History Table

The FlyerHistory Table’s purpose is to store all parameters entered by a Store Manager when they generate a flyer. This is to keep a company record of flyers created and to allow a district manager the ability to see what flyers have been created.

These values will be read out by the program to generate the exact flyer the Manager used instead of saving the entire flyer image onto the hard drive, saving hard drive storage space and keeping the data better protected.

Each FlyerHistory Table entry will contain the following fields:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Allow Nulls |
| ID (Primary Key, Unique) | Int | No |
| Manager Name | Text | No |
| Template Name | Text | No |
| Store Address | Text | No |
| Store Phone Number | Text | No |
| Flyer Creation Date | Text | No |
| Flyer Sale Dates | Text | No |
| Item 1 Name | Text | No |
| Item 1 Price | Int | No |
| Item 1 Image | Text | No |
| Item 2 Name | Text | Yes |
| Item 2 Price | Int | Yes |
| Item 2 Image | Text | Yes |

To save excessive reading the Item X Name, Price and Image fields will continue using the Item 2 layout until Item 15 as per SRS

* ID (Primary Key, Unique) – Unique Identifier for each flyer created
* Manager Name – This is the Name and ID number of the manager that created the flyer
* Template Name – This is the template the Manager used to create the flyer. This is needed for the Flyer Creator program to find the correct template type
* Flyer Creation Date – This is the date the flyer was created. This will be used in the Flyer History program to allow a District Manager to view all flyers created on a specified date
* Flyer Sale Dates – This will contain the dates the Manager set for the flyer
* Item Name – These fields will be used to populate the item name in the recreated flyer
* Item Image – These fields will be used to populate the item image in the recreated flyer
* Item Price – These fields will be used to populate the item price in the recreated flyer

### Section 3.4.2 Item List Table

The ItemList Table’s purpose is to store the vast amount of products Food Giant sells. This data will be pulled out of the database to allow a manager the ability to select the item they want, and display the images associated with the item.

The ItemList Table will contain the following fields:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Allow Nulls |
| ID (Primary Key, Unique) | Int | No |
| Item Name (Unique) | Text | No |
| Item Category | Varchar (50) | No |
| Image Name 1 | Text | No |
| Image Name 2 | Text | Yes |
| Image Name 3 | Text | Yes |

ID – Unique Identifier for each Food Giant Inventory database item

Item Name – Unique Identifier for each Food Giant Inventory database item

Item Category – Category to group each item in. Will be used to quickly view number of items for each category, which is useful for data gathering

Image Name - These fields handle the images associated for each item. The second and third item can be null, since there may only be one required image for the item entered.

The Item’s Price will be entered by Manager in the Flyer Creator Program. This is because the client stated that the price can vary dramatically from one region to another so it will be up to the Store Manager to set the proper sale price for the item, instead of giving them a default price to change.

## Section 3.5 Database Interface

This class satisfies requirement SR 2.3.

This class will contain all the database queries used by the application. Database Interface will be the **only** class that contains queries to connect database. This is to enforce a secure, central location to retrieve data from the database.

This class will contain the following methods:

* SQLCommand CreateConnectionString
* SQLCommand GetInventoryItems
* SQLCommand EnterNewInventoryItem
* SQLCommand UpdateInventoryItem
* SQLCommand GetFlyerTemplateHistory
* Void ConnectToDatabase
* Void RunSQLCommand

All connects to the database will go through the Connect to Database method, which will contain all error handling and Query Validation. This is to have a single protected entry point into the database reducing the chance of damaging the database from a malformed query.

## Section 3.6 Database Maintainer Program Classes

### Section 3.6.1 Overview of Database Maintainer Program Classes

This class satisfies requirement SR 2.4.

The Database Maintainer classes purpose is to allow a manager or manager’s assistant to add new items to the SQL Database. Since neither person will be expected to have any database experience, we will need to create these classes to simplify the process and add in error handling to reduce the chance of a bad database entry.

This class will also back-up the databases when a manager or assistant finishes making database updates. This will be another safety net to further reduce the chance of data loss.

### Section 3.6.2 View

The Database Maintainer’s View will contain a very similar layout to the Item Name, Price, and Image section of the Flyer Creator View. The difference being that the DM View will only allow one entry at a time, and have a log below the entry fields allowing the manager to see what actions have been done. This is important if the manager is expecting to enter in a significant amount of data entries.

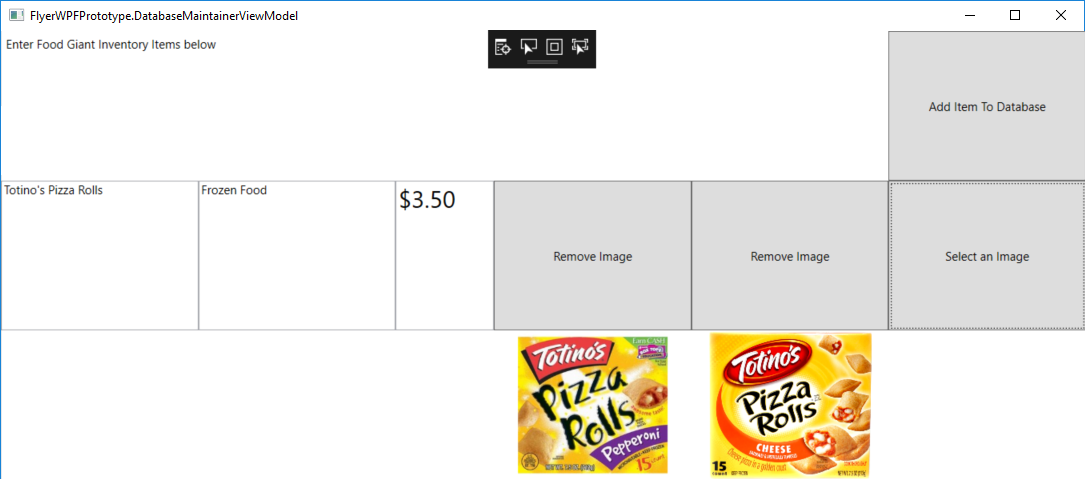


Image 1.2 Database Maintainer Prototype Picture

### Section 3.6.3 ViewModel

The View Model will contain all the event handlers and validation logic for the program. This class will validate all options selected by the Manager and connect to DBInterface to save the entered item into the SQL Database.

## Section 3.7 FlyerDataModel

This model will contain the variables needed to create a new Food Giant Inventory item. This model will be used in the Database Maintainer Class for when the manager adds new items to the database to validate the items before entered in the database query, and to allow the Flyer Creator classes to pull out the individual values to populate the data fields.

This class will consist of the following values:

* public string itemName
* public string itemCategory
* public string itemPrice
* public string imageName1
* public string imageName2
* public string imageName3

## Section 3.8 Program Selector

This program satisfies requirement SR 3.1

This program will validate the client’s credentials and only start up if they are a District Manager allow them to select the program they wish to start.

This program will exist as a Window containing a Welcome message and a simple drop down menu containing the following options:

* Create a new Flyer
* View an Existing Flyer
* Add a new Grocery Item

The program will simply open the requested program in a new Window. This program will continue to run in the background so once a Manager finishes their task they can go back to the Program Selector to start a new program. They cannot start a new program while they have an existing program running.

# Section 4 Terms of Reference

|  |  |
| --- | --- |
| Term | Definition |
| ASP.NET | Active Server Pages |
| ASPX | Active Server Page Extended |
| C# | C-Sharp Programming Language |
| GUI | Graphical User Interface |
| MVVM | Model View View-Model |
| QA | Quality Assurance |
| SDD | Software Design Description |
| SLN | Visual Studio Solution |
| SPMP | Software Project Management Plan |
| SQL | Structured Query Language |
| SRS | Software Requirements Specifications |
| STD | Software Test Document |
| VS | Visual Studio |
| XAML | Extensible Application Markup Language |