# Final Project – Price Comparer System - Documentation

**Table of Contents**

1. Requirements
   1. Basic Requirements
   2. Extended Requirements (Bonuses)
2. System Components
   1. Diagram
   2. Feeders
      1. UI
   3. Managers
      1. CatalogManager (+interface: ICatalogManager)
      2. CartManager(+interface)
      3. PriceComparisonManager(+interface)
      4. NotificationManager
   4. Engines
      1. CatalogEngine
      2. CartEngine

*My design was based on the IDesign method produced by Juval Lawey.*

1. **Requirements**

**1.1 Basic Requirements**

* Decode XML files of chain stores taken from a governmental web site (XElement and LINQ)
* Store a catalog and manage it catalog, items (products) and their prices
* Enable users to build a cart of items and quantities
* Enable users to compare cost of a cart between different chain stores
* Produce final cost of a cart for each chain store, and so the three cheapest items and three most expensive ones

**1.2 Extended Requirements (Bonuses)**

1. Add GUI(Windows forms application)
2. Enable users to compare prices according to geographic locations (cities), specific chains or specific stores
3. Enable users to save their cart in a file (xml), and load it later
4. Store and manage a catalog inside a database (Microsoft SQL Server 2016)
5. ???
6. Manage users and their cart inside the database
7. Separate the system – Client-Server – asp.NET web application
8. Produce an Excel file with price comparison chart and graph
9. Produce a graph describing a price of a chosen item of a chosen store changing through time
10. **System Components**

**2.1 Diagram – blocks and arrows**

**2.2 UI**

* The UI is implemented as a Window Forms Application.

**2.3 ICatalogManager**

**Interface Methods**

* async AddChainToCatalog(Chain)
* async AddStoreToCatalog(Store)
* async AddItemToCatalog(Item)
* async AddPriceToCatalog(Store, Item)
* async GetItemsByName(string name) – Free search
* async GetItemsByStore(Store)
* async GetAllChains()
* async GetAllStoresByChain(Chain)
* async UpdateCatalogFromXmlFiles()

**2.4 ICartManager**

* async AddItemToCart(Cart, Item)
* async RemoveItemFromCart(Cart, Item)
* async CalculateCartTotalCost(Cart, Store)

**2.5 IPriceComparisonManager**

* async CompareCarts(ICollection<Cart>)

**2.6 UIManager (Maybe)**

* AddToCartButtonClick()

**Example**

Form1.btnAddToCartl\_click()UI.btnAddItemToCartl\_click()  CartManager.AddItemToCart(Cart, Item)

**IMPORTANT – each method in any manager should be invoked with Task.**

**INotificationManager**

Call notification for the user using prompts with message boxes.

Interface Methods

* NotifyExitApplication()
* NotifyCatalogUpdated()
* NotifyItemInsertedIntoCart()
* NotifyItemRemovedFromCart()
* NotifyBadSearchStringForItem()

**CatalogEngine**

This engine should implement Catalog manager methods using LINQ queries, and it should use DatabaseConnector to access the catalog database.

**Example:**

Form1.btnAddToCartl\_click()UI.btnAddItemToCartl\_click()  CartManager.AddItemToCart(Cart, Item)  CartEngine.AddItemToCart(Cart, Item)

**Use cases**

**Modules**

**Classes**

* MainForm
* DatabaseConnector
* XmlDecoder
* Catalog
* Basket
* Item
* Chain
* Store
* Promo

**Extensions and Bonuses**

* Graphic User Interface

For simplicity and convenience, I chose to use Windows forms application.

* Products Catalog management using database

I chose to use MySQL database to manager the products catalog, because I have an experience with this kind of databases, and its workbench is very simple.