POS SYSTEM – ARCHITECT DESIGN DOCUMENT



HIT Team

Consulting

Sales

Staffing

Support

Information of document

|  |  |
| --- | --- |
| Title | Architecture Driver Document |
| Author(s) | All team |
| Reviewer(s) | All team |
| Team name | HIT Team |
| Team members | Thanh Giang, Hiep Ta, Phuc Nguyen, Giang Nguyen, Dat Tran, Huy Huynh |
| Project mentors | Ms. Huong Nguyen, Mr. Huyen Pham |
| Editor | Hiep Ta |
| Type of report | Architecture Driver Document |
| Software used | MS Word |

Document Reviewer Information

|  |  |  |
| --- | --- | --- |
| **Reviewer Name** | **Review Attendance (R/S)** | **Comments** |
|  |  |  |
|  |  |  |
|  |  |  |

Document Approver Information

|  |  |  |
| --- | --- | --- |
| **Approver Name** | **Approver Function** | **Comments** |
|  |  |  |
|  |  |  |
|  |  |  |

Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Date of Issue | Author(s) | Brief Description of Change |
| 1.0 | 06/03/2012 | Thanh Giang | Consume Team member’s tasks |
| 1.1 | 15/04/2012 | Dat Tran | Update Quality Attributes |
| 1.2 | 18/04/2012 | Thanh Giang | Use case |
| 2.0 | 24/05/2012 | Hiep Ta | Use case Description |
| 2.1 | 25/5/2012 | Dat Tran | Sequence |
| 2.2 | 28/5/2012 | Trong Giang | Coding |
| 3.1 | 09/06/2012 | Huy Huynh | Module View |
| 3.2 | 20/07/2012 | Thanh Giang | Allocation View, Data Model, Context Diagram, |
| 3.3 | 21/06/2012 | Huy Huynh | Static Perspective & Dynamic Perspective |
| 3.4 | 5/07/2012 | Thanh Giang, Huy Huynh | Mapping View |

Table of Contents

[1. Document description 4](#_Toc329248612)

[1.1. Purpose and audience: 4](#_Toc329248613)

[1.2. Document organization: 4](#_Toc329248614)

[1.3. Common notation: 4](#_Toc329248615)

[1.4. Terminology and definitions: 4](#_Toc329248616)

[1.5. References and relevant document: 5](#_Toc329248617)

[2. Project overview 6](#_Toc329248618)

[2.1. Project Context 6](#_Toc329248619)

[2.2. System Context 6](#_Toc329248620)

[3. Architectural drivers 8](#_Toc329248621)

[4. System context 9](#_Toc329248622)

[5. Level X decomposition 10](#_Toc329248623)

[6. Mapping between perspectives 11](#_Toc329248624)

[7. Document directory 12](#_Toc329248625)

# Document description

## Purpose

Intending to capture and convey the architectural decisions that have been made in order to implement POS system, the Software Architecture Design Document (SADD) formally provides a comprehensive overview of the proposed system. It uses a number of architectural decompositions to depict the different aspects, corresponding with the architecture driver specified. This document serves as a basis for the detailed design, which will establish the design in increased detail.

## Audience

The audiences for this document include:

Supervisor

|  |  |
| --- | --- |
| Name | E-mail |
| Mrs. Nguyen Thuy Huong | ngthuong02@yahoo.com |

HIT Team

|  |  |
| --- | --- |
| Name | E-mail |
| Giang Thi Ha Thanh | giangthihathanh@gmail.com |
| Ta Quang Hiep | hiep.taquang@gmail.com |
| Huynh Chan Huy | huynhchanhuy@gmail.com |
| Nguyen Tran Hong Phuc | phucnguyen10121991@gmail.com |
| Tran Dung Dat | dattran2525@gmail.com |
| Nguyen Vu Trong Giang | chan.tinh157@gmail.com |

Administrators

Administrators of POS system may use this design to understand the structure of the system.

Future Developers of the product

This document is written such that any future developers employed for enhancements or modifications to the POS system code may use it to understand the existing system.

## Terminology and definitions:

|  |  |
| --- | --- |
| ACDM | Architecture Centric Design Method |
| Mapping | A mapping is created either between different views or between elements or relationships and groups of elements and relationships. A mapping can be hierarchical, or on the same level. Mappings between elements or relationships and groups of elements and relationships can span across views. |
| Entity | The ACDM definition for an actor which could be a person or a system. |
| Relationship Type | A relationship type is a connector in the palette, which is used to create a relationship in the canvas. |
| Relationship | A relationship is an instance of a relationship type, and used to connect element(s) on the canvas. |
| Perspective | 3 perspectives prescribed by ACDM; static, dynamic and physical. |
| View | A view is a drawing in a perspective. There can be multiple views in a perspective. |
| Design | This is the drawing the architect does in the tool. The term design is used in this document because architecture is used in many other ways which might confuse the reader. |
| Architecture | All of the diagrams including their elements and relationships as a representation of a software system. The architecture also includes the architectural drivers which are assigned to the elements and relationships. |
| Architectural drivers | These consist of functional requirements, quality attributes, business constraints and technical constraints. Can be assigned to elements, relationships and mappings. |
|  |  |

## References and relevant document:

### References

[Sandcastle] - Architectural Drivers Specification

[Lat08] Lattanze, A. *Architecting Software Intensive Systems: A Practitioners Handbook*, New York, NY: Auerbach, 2008

Software Architecture Design Document - Collaborative Problem Solver

### Relevant document

POS System – Architecture Driver Document

# Project overview

* This section describes the project and its purpose and scope.
* Describe why the system is being built.
* Describe business, organizational, mission that are relevant to the project.
* List the relevant stakeholders, their organizations, and how they will interact with the system.

## Project Context

It is Final project that requires HIT Team to develop POS System with the following constrains:

* Team size: 6 members
* The time for completing: 3 months
* Software Development Model assigned: Water fall
* Tools assigned: ASP .NET Framework, MS SQL 2008, Visual studio 2010.
* Hardware: Laptops

## System Context

The project team will develop a sale system on websites with the purpose bring the convenience and helpful in the sale for Company A, a retail chain (hereinafter, the system). POS system will contain a number of functions that help manage product as well as sale process.

* Administrator can do following function:
* Add New User
* Search/ View User List
* View User Detail Information
* Update User Information, Assign Authorize
* Add New POST
* Search/ View POST List
* View POST Detail Information
* Update POST Information
* Sync Information
* Staff can do following function:
* Add New Product
* Search/ View Product List
* View Product Detail Information
* Update Product Information
* Add New Type
* Search/ View Type List
* View Type Detail Information
* Add New Member
* Search/ View Member List
* View Member Detail Information
* Update Member Information
* View Member Point Log
* Cashier can do following function:
* Add New Store
* Search/ View Store List
* View Store Detail Information
* Update Store Information
* Analysis Statistic
* Cashier can do following function:
* Add New Bill
* Search/ View Bill List
* View Bill Detail Information
* Print Bill
* Member can do following function:
* View point

# Architectural drivers

These architectural drivers will influence the architectural design and implementation of the project. Additionally, they will impact the schedule and quality of the project. As a whole these architectural drivers define the scope of the project.

The architectural drivers presented in this document include:

* **Functional Requirements:** These requirements are presented in the form of specifications and use cases. These are a refinement of the requirements documented in the raw requirements specification document of step 1 ACDM.
* **Quality Attribute Requirements:** These requirements are presented in the form of quality attribute scenarios. These scenarios are based on the quality attributes documented in the raw requirements specification document of step 1 ACDM.
* **Business Constraints:** These are the business constraints documented in the raw requirements specification document of step 1 ACDM.
* **Technical Constraints:** These are the technical constraints documented in the raw requirements specification document of step 1 ACDM.

Detail architecture driver please refer to file: POS System – Architecture Driver

# Architecture Design

## Style

We use client-server style because of reasons:

* Centralised resources: given that the server is the centre of the network, it can manage resources that are common to all users, for example: a central database would be used to avoid problems caused by redundant and inconsistent data
* Improved security: as the number of entry points giving access to data is not so important.
* Scalable network: thanks to this architecture it is possible to remove or add clients without affecting the operation of the network and without the need for major modification.
* Centralized Storage Resources exist on the server and can be shared by all clients. Instead of clients redundantly storing information, a single version is stored on the server for clients to access.

## Framework

### Entity Framework

In model, we use entity framework because its benefit:

* + With Entity Framework, the developers issue queries using LINQ, then retrieve and manipulate data as strongly typed objects. The Entity Framework’s ORM implementation provides services like change tracking, identity resolution, lazy loading, and query translation so that developers can focus on their application-specific business logic rather than the data access fundamentals.
  + Its goal is to decrease the amount of code and maintenance required for data-oriented applications. Entity Framework applications provide the following benefits:
    - Applications can work in terms of a more application-centric conceptual model, including types with inheritance, complex members, and relationships.
    - Applications are freed from hard-coded dependencies on a particular data engine or storage schema.
    - Mappings between the conceptual model and the storage-specific schema can change without changing the application code.
    - Developers can work with a consistent application object model that can be mapped to various storage schemas, possibly implemented in different database management systems.
    - Multiple conceptual models can be mapped to a single storage schema.
    - Language-integrated query (LINQ) support provides compile-time syntax validation for queries against a conceptual model.

### MVC Framework

There are 3 components in MVC framework: View, Model, and Controller

* The controller in MVC undertakes tasks that are implemented on Business Logic Layer and Data Access Layer. Moreover, View render a webpage to client as a UI and controller can support a view by returning appropriate view to client base on its request.
* On Web environment, increasing performance is important. Therefore, eliminating a layer is needed.
* Also, the scope of system isn’t large and doesn’t require much more complex business logic calculations, thus, MVC framework can fulfill its job.

Let’s get to the bottom of MVC framework:

* The view is responsible for providing the user interface (UI) to the user. It is given a reference to the model, and it transforms that model into a format ready to be presented to the user.
* The controller is responsible for responding to user input, often making changes to the model in response to user input. In this way, controllers in the MVC pattern are concerned with the flow of the application, working with data coming in, and providing data going out to the relevant view.
* The model that is used to send information to the Data Access Layer, performs business calculations, and even render in a view. Otherwise, these objects represent the domain of the application focuses on, and the models are the objects you want to save, create, update, and delete.

There is the list of MVC# framework features that is the reason why we use this:

* Views and controllers get connected automatically. The MVC framework automatically establishes links between views and corresponding controllers. Therefore, the developers do not care about the associating views and controllers that linked to their views.
* Tasks concept. Sometime, we have to unite several views with their controllers to do some job, this is called a task. For example a checkout task may consists of two views, one to choose a product (such as: Milk, Drink, Cake… we can order in supermarket), the other – to do the payment. In MVC# all controllers within a task are given a link to the task object. Generally a task can be expressed as a workflow or a state machine.
* It is easier to test components when they do not rely upon one another.
* Because the application is easily to testing, so it make an application is more scalable.

# System context



## Users and roles:

Stakeholders who interact to Retail system were described in section 2.3.Stakeholder of this document.

## Channels:

Users will use different channels to access the system.

* Staff: Access website to system by PC locates at Head Office.
* Manager: Access website to system by every PC.
* Cashier: Use Bar code reader at computer using POS website to interacts with system
* Customer: Use online website to see their point
* Administrator: Use PC at Head Office to access the system

## Relationship Describe:

The context diagram shows the input of stakeholders and output from system, direction of the arrows show the direction of information.

: Show that the input from user to the system

: Show that the output from system to the user

# Physic Perspective

**ALLOCATION VIEW (Deployment Style)**

## Primary presentation



## Element catalog:

### Elements and their properties

.

|  |  |  |  |
| --- | --- | --- | --- |
| Associated Drawings:  Fig2 | | | Perspective:  Physic |
| No | **Name** | **Properties** | **Responsibilities** |
| 1 | Main Database | * It‘s a database run in database management system SQL server 2008 | * Contains all general data of system |
| 2 | Backup database | * It‘s a database run in database management system SQL server 2008 | * Contains all general data of system. Run parallel with the main database |
| 3 | WEB POS Application | * It’s a WEB application |  |
| 5 | Database Server | * OS: Windows Server 2008 * Processor: 1 x Intel® Xeon® Processor E5606 * Memory: 1 x 2GB DDR3 1333 240- * Hard Disk: DELL 250GB SATA 7.2K 3.0Gbs 3.5" Enterprise * Software: Microsoft SQL Server 2008 Enterprise, .NET Framework 4.0 | * Run Main Database * Run Backup Database |
| 6 | WEB server | * Software: .Net Framework 4, IIS 7 |  |
| 6 | User PC | * Operation System : Genuine Windows® 7 Home Basic, * Processor: AMD AM3 For Phenom™ II/Athlon™ II Family /Processors * Chipset: AMD SB710 * Graphics: ATI Radeon HD 3200, * Hard Drive SATA: 3.5" 320G * Software: Web browser | * Head office and Retail Store   + Run WEB POS Application |
| 7 | Fiber router | * Vigor2950 | * Connect LAN with WAN |
| 8 | LAN | * Topology: Star * Use switch to connect elements in LAN | * Connect computers in a store |
| 11 | Fiber cable |  | * Connect fiber router in WAN by fiber port |

## Architecture background

### Rationale design

Firstly, we have to define some “technique” that we will use to design allocation view

* **A server cluster** is a group of independent servers running Windows Server 2003 Datacenter Edition, and working together as a single system to provide high availability of services for clients. When a failure occurs on one computer in a cluster, resources redirected and the workload redistributed to another computer in the cluster. We use server clusters to ensure that users have constant access to important server-based resources. And database have a process which is similar as server
* **Load balancing** is the process by which inbound internet protocol (IP) traffic can distributed across multiple servers. Load balancing enhances the performance of the servers, leads to their optimal utilization and ensures that no single server is overwhelmed

Deployment design satisfies quality attributes following:

* + **Performance**: We use load balancing (inbound internet protocol (IP) traffic) to disperse the requests, balance the requirements for the two servers that ensure the performance and availability of the system
  + **Security**: The Firewall will prevent unauthorized or unwanted communications so that we can save database and web server. POS Database is placed in internal network, not with Webserver which placed in DMZ network.
  + **Availability**: Our system has two databases (a server cluster) to be able to process multiple requests: maintain multiple copies of either data or computations all data store in database server and two web servers: disperse the requests

# Static Perspective

**Module View (Layered Style and Uses Style)**

## Primary presentation:

### A combined Decomposition – Layer view



### Uses Style

* + - 1. View Packet 1: Sale Management



* + - 1. View Packet 2: Store Management



* + - 1. View Packet 3: Customer Management



* + - 1. View Packet 4: Computer Management



* + - 1. View Packet 5: User Management



* + - 1. View Packet 6: Product Management



* + - 1. View Packet 7: Category Management



* + - 1. View Packet 8: Statistics Management



## Element catalog:

### Elements and their properties

|  |  |  |
| --- | --- | --- |
| Elements | | Properties |
| Controller | **ProductControllers** | The ProductController class contains action methods that render view pages (AddProduct, EditProduct, ViewProduct) |
| **Category**  **Controllers** | The CategoryController class contains action methods that render view pages (AddCategory, EditCategory, ViewCategory) |
| **StatisticsControllers** | The StatisticsController class contains action methods that render view pages (Statistics) |
| **SaleControllers** | The SaleController class contains action methods that render view pages (BillManagement, Checkout, PriceLog) |
| **StoreControllers** | The StoreController class contains action methods that render view pages (AddStore, ViewStore, EditStore) |
| **LoyalMember**  **Controllers** | The LoyalMemberController class contains action methods that render view pages (AddLoyalMember, ViewLoyalMember, EditLoyalMember) |
| **StoreCategory**  **Controllers** | The StoreCategoryController class contains action methods that render view pages (AddStoreCategory, ViewStoreCategory, EditStoreCategory) |
| **ComputerControllers** | The POSController class contains action methods that render view pages (AddPOS, ViewPOS, EditPOS) |
| **UserControllers** | The UserController class contains action methods that render view pages (AddUser, ViewUser, EditUser) |
| View | **AddStore** | This GUI helps the user to add a new Store. |
| **ViewStore** | This GUI helps the user to view the Store. |
| **EditStore** | This GUI helps the user to modify some information about the Store. |
| **AddCategory** | This GUI helps the user to add a new Product Category. |
| **ViewCategory** | This GUI helps the user to view the Product Category. |
| **EditCategory** | This GUI helps the user to modify some information about the Product Category. |
| **AddProduct** | This GUI helps the user to add a new Product. |
| **ViewProduct** | This GUI helps the user to view the Product. |
| **EditProduct** | This GUI helps the user to modify some information about the Product. |
| **BillManagement** | This GUI helps the user to view the list of Bills and the user can see some detail information of each Bill. |
| **AddUser** | This GUI helps the user to add a new User. |
| **ViewUser** | This GUI helps the user to view the User. |
| **EditUser** | This GUI helps user to modify some information about the User. |
| **AddLoyalMember** | This GUI helps the user to add a new loyal Customer. |
| **ViewLoyalMember** | This GUI helps the user to view the loyal Customer. |
| **EditLoyalMember** | This GUI helps user to modify some information about the loyal Customer. |
| **AddStoreCategory** | This GUI helps the user to add a new Retail Store Category. |
| **ViewStoreCategory** | This GUI helps the user to view the Retail Store Category. |
| **EditStoreCategory** | This GUI helps user to modify some information about the Retail Store Category. |
| **AddComputer** | This GUI helps the user to add a new Computer. |
| **EditComputer** | This GUI helps user to modify some information about the Computer. |
| **ViewComputer** | This GUI helps the user to view the Computer. |
| **Statistics** | To make statistics about the total amount of product (or product category) was bought on month. |
| Model | **Product** | Storing and retrieving the Product information and return a message back to view pages (AddProduct, EditProduct, ViewProduct) |
| **Bill** | Storing and retrieving the Sale information and return a message back to view pages (BillManagement) |
| **Bill\_Detail** | Storing and retrieving the Sale information and return a message back to view pages (BillManagement) |
| **RetailStore** | Storing and retrieving the Retail Store information and return a message back to view pages (AddStore, EditStore, ViewStore) |
| **Customer** | Storing and retrieving the Customer information and return a message back to view pages (AddCustomer, EditCustomer, ViewCustomer) |
| **User** | Storing and retrieving the Userinformation and return a message back to view pages (AddUser, EditUser, ViewUser) |
| **Category** | Storing and retrieving the Product Categoryinformation and return a message back to view pages (AddProductCategory, EditProductCategory, ViewProductCategory) |
| **RetailStore\_Category** | Storing and retrieving the Retail Store and Category information and return a message back to view pages. |
| **Computer** | Storing and retrieving the Computerinformation and return a message back to view pages (AddComputer, EditComputer, ViewComputer) |
| **Cost** | Storing and retrieving the Product and RetailStore information and return a message back to view pages. |
| **DBContext** | Context class is the primary class for interacting with data as objects that are instances of entity types that are defined in an Entity Data Model (EDM). |

* + 1. Relations and their properties

|  |  |
| --- | --- |
| Connector | Properties |
| Allowed to use | The layers are related to each other by the strictly ordered relation allowed to use. |
| Uses | The uses style shows how modules depend on each other; it is helpful for planning because it helps define subsets and increments of the system being developed. |

## Architecture background:

The application divided into 3 layers: Model, View, and Controller.

There are reasons that decision design meets the requirement will be explained as below:

* We use entity framework work with database server (Microsoft SQL Server). It can handle real-world database schemas and works well with stored procedures. If database changed, it will not affect controller, ensuring mobility attribute.
* To improve performance for system, we uses output caching method. You can dramatically improve the performance of ASP.NET MVC web applications by taking advantage of output caching.
  + The output cache allows you to cache the content returned by a controller action. The same content does not need to be generated each and every time the same controller action is invoked.
  + Taking advantage of the output cache and then you can avoid executing a database query every time any user invokes the same controller action. The view can be retrieved from the cache instead of being regenerated from the controller action. Caching enables you to avoid performing redundant work on the server.

# Data Model

## Primary presentation:



## Element Catalog

### Elements and their properties

|  |  |  |  |
| --- | --- | --- | --- |
| Entity | Attributes | Data Type | Description |
| Bill | **BillID** | Varchar(9) |  |
| **POSTMAC** | VARCHAR(17) | Attribute said bill is made in which Computer well as the general store |
| **CustomerID** | Varchar(9) | Customer pays the invoice. |
| **UserID** | Varchar(9) | Cashier |
| TotalCost | FLOAT | The total cost of the bill, ensuring the implementation of Statistical Performance Data |
| Date | DATETIME | Paid Bill Date |
| PlusPoint | INT | Minus and plus points in a session will be stored here. |
| MinusPoint | INT |
| Product | **ProductID** | Varchar(9) |  |
| Product\_Name | NVARCHAR(50) |  |
| BasicCost | FLOAT | AttributeBasic Cost show the default price of the product |
| **CategoryID** | Varchar(9) |  |
| Stock |  | The merchandise that a shop has on hand |
| PrStatus | BIT |  |
| Bill\_Detail | **BillID** | Varchar(9) |  |
| **ProductID** | Varchar(9) |  |
| Quantity | INT | Sum Loyal Point of customer |
| Prize | INT |  |
| Customer | **CustomerID** | Varchar(9) |  |
| Customer\_Name | NVARCHAR(50) |  |
| Customer\_Address | NVARCHAR(50) |  |
| Customer\_Phone | VARCHAR(15) |  |
| SumPoint | INT | Sum Loyal Point of customer |
| CuStatus | BIT |  |
| RetailStore | **RetailStoreID** | Varchar(9) |  |
| RetailStore\_Name | NVARCHAR(50) |  |
| ReStatus | BIT |  |
| Cost | **ProductID** | Varchar(9) | Entity Cost said that Retail Store Retail Store selling a certain product and pricing individual products within a certain time. |
| **RetailStoreID** | Varchar(9) |
| DateStart | DATETIME |
| DateEnd | DATETIME |
| Cost | FLOAT |
| Category | **CategoryID** | Varchar(9) |  |
| Category\_Name | NVARCHAR(50) |  |
| CaStatus | BIT |  |
| RetailStore\_Category | **RetailStoreID** | Varchar(9) |  |
| **CategoryID** | Varchar(9) |  |
| Quantity | INT |  |
| POST | **POSTMAC** | VARCHAR(17) |  |
| **RetailStoreID** | Varchar(9) | This Attribute tells us this POS Terminal is placed at which Retail Store |
| CoStatus | BIT |  |
| User | **UserID** | Varchar(9) |  |
| User\_Name | NVARCHAR(50) |  |
| User\_Address | NVARCHAR(50) |  |
| User\_Phone | VARCHAR(15) |  |
| Password | VARCHAR(32) |  |
| **RetailStoreID** | Varchar(9) | This Attribute tells us this user works at which Retail Store |
| UsStatus | BIT |  |

## Architecture Background

* Data model design satisfy business constrains following:
  + Not all stores carry every product type, and the range of product types carried is designated for each store.
  + Can set and use its own actual retail price instead of the standard price during the limited period specified be each store.

# Dynamic Perspective

**Component and Connector View**

## Primary presentation



## Element catalog:

### Elements and their properties

|  |  |  |
| --- | --- | --- |
| Elements | | Properties |
| Client tier | **Web Browser** | A component that sends a request to web server and receive a reply is sent by web server. And then it displays UI to user. |
|  | **IIS 7** | IIS 7 is a major enhancement to the Windows web platform and plays a central role in unifying Microsoft web platform technologies - ASP.NET, Windows Communication Foundation web services, and Windows SharePoint Services.  It handles requests and replies that are send between client tier and server tier. |
| Server Tier | **ProductControllers** | The ProductController class contains action methods that render view pages (AddProduct, EditProduct, ViewProduct) |
| **Category**  **Controllers** | The CategoryController class contains action methods that render view pages (AddCategory, EditCategory, ViewCategory) |
| **StatisticsControllers** | The StatisticsController class contains action methods that render view pages (Statistics) |
| **SaleControllers** | The SaleController class contains action methods that render view pages (BillManagement, Checkout, PriceLog) |
| **StoreControllers** | The StoreController class contains action methods that render view pages (AddStore, ViewStore, EditStore) |
| **CustomerControllers** | The LoyalMemberController class contains action methods that render view pages (AddLoyalMember, ViewLoyalMember, EditLoyalMember) |
| **ComputerControllers** | The POSController class contains action methods that render view pages (AddComputer, ViewComputer, EditComputer) |
| **UserControllers** | The UserController class contains action methods that render view pages (AddUser, ViewUser, EditUser) |
| **AddStore** | This GUI helps the user to add a new Store. |
| **ViewStore** | This GUI helps the user to view the Store. |
| **EditStore** | This GUI helps the user to modify some information about the Store. |
| **AddCategory** | This GUI helps the user to add a new Product Category. |
| **ViewCategory** | This GUI helps the user to view the Product Category. |
| **EditCategory** | This GUI helps the user to modify some information about the Product Category. |
| **AddProduct** | This GUI helps the user to add a new Product. |
| **ViewProduct** | This GUI helps the user to view the Product. |
| **EditProduct** | This GUI helps the user to modify some information about the Product. |
| **BillManagement** | This GUI helps the user to view the list of Bills and the user can see some detail information of each Bill. |
| **AddUser** | This GUI helps the user to add a new User. |
| **ViewUser** | This GUI helps the user to view the User. |
| **EditUser** | This GUI helps user to modify some information about the User. |
| **AddCustomer** | This GUI helps the user to add a new loyal Customer. |
| **ViewCustomer** | This GUI helps the user to view the loyal Customer. |
| **EditCustomer** | This GUI helps user to modify some information about the loyal Customer. |
| **AddComputer** | This GUI helps the user to add a new Computer. |
| **EditComputer** | This GUI helps user to modify some information about the Computer. |
| **ViewComputer** | This GUI helps the user to view the Computer. |
| **Statistics** | To make statistics about the total amount of product (or product category) was bought on month. |
| **Product** | Storing and retrieving the Product information and return a message back to view pages (AddProduct, EditProduct, ViewProduct) |
| **Bill** | Storing and retrieving the Sale information and return a message back to view pages (BillManagement) |
| **Bill\_Detail** | Storing and retrieving the Sale information and return a message back to view pages (BillManagement) |
| **RetailStore** | Storing and retrieving the Retail Store information and return a message back to view pages (AddStore, EditStore, ViewStore) |
| **Customer** | Storing and retrieving the Customer information and return a message back to view pages (AddLoyalMember, EditLoyalMember, ViewCustomer) |
| **User** | Storing and retrieving the Userinformation and return a message back to view pages (AddUser, EditUser, ViewUser) |
| **Category** | Storing and retrieving the Product Categoryinformation and return a message back to view pages (AddProductCategory, EditProductCategory, ViewProductCategory) |
| **RetailStore\_Category** | Storing and retrieving the Store Category information and return a message back to view pages |
| **Computer** | Storing and retrieving the POSinformation and return a message back to view pages (AddComputer, EditComputer, ViewComputer) |
| **Cost** | Storing and retrieving the Product and RetailStore information and return a message back to view pages. |
| Database tier | **Primary Database** | It’s main database server which uses frequently when the system work properly. It’s responsible for store data such as sales data, user data, customer data, store data, product data, and category data. In the certain time, it will synch with temporary database as a backup data. |
| **Backup Database** | It’s a temporary database which uses rarely. It only uses when the system doesn’t work properly, crash or not available. In the certain time, it will synch all data with primary. |

### Relations and their properties

|  |  |
| --- | --- |
| Connector | Properties |
| Request/ Reply | Connector between client and server style, used by a client to invoke services on a server. |
| Call and return | Responsible for conveying the service request from the requester to the provider and for returning any results. Use by interface to request data from Filter/Object |
| Synchronous Replication | The technique for replicating data by two or more databases (or file systems) where the system being replicated does waits for the data to have been recorded on the duplicate system before proceeding. |
| ODBC | Stands for Open Database Connectivity. It is the standard method which allows any application to connect data. ODBC uses a middle layer called the database driver to handle the connection in between the application and the relational database management system. |

## Architecture Background

The application divided into 3 tier: client tier, server tier, database tier.

* On database tier, we have a cluster contain two databases and those databases will sync with each other in an hour. If one of those database is crash, another one will active, ensuring system availability.
* We use entity framework to work with database server (Microsoft SQL Server). It can handle real-world database schemas and works well with stored procedures. It is mapping between the database schema and the conceptual schema, load data (tables, stored procedures, views) from Database to memory. Afterward, it loads data to implement client’s request from memory instead of database server, increasing system performance.

The reason that our design meets quality attributes that will be explained in the following table

|  |  |  |  |
| --- | --- | --- | --- |
| **Quality** | **Quality ID** | **Concern** | **Response measure** |
| Performance | QAS3 | Displays statistical reports | The system get information in primary database and response to client within 5 seconds |
| **Rationale:** To know a business status, the manager performs the statistical reports, the system displays information reported within 5 seconds | | | |
| Performance | QAS4 | Scan Loyal Member | The system get information in primary database and response to client within 3 seconds |
| **Rationale**: When the member go to the store, choose some product and go to the POS to pay a bill. The cashier scans the member's card; the system displays the Member information within 3 seconds | | | |
| Performance | QAS2 | Bill | The system save information in primary database and response to client save bill success within 5 seconds |
| **Rationale:** The cashier sends confirmation of payment request, system confirm request, save billing information in the primary database and notify successful payment within 5s | | | |
| Availability | QAS1 | Sale product when database at Head Office crash | The system gets information in backup database and response to client within 5 second. |
| **Rationale:** When the primary database in the Head Office has been crash. The cluster exchange database, backup database will use instead primary database and response to client within 5 second. | | | |
| Security | QAS5 | Block unauthorized access |  |
| **Rationale:** When the users try to access the system illegitimacy, the system block this request | | | |
| Security | QAS6 | Authority |  |
| **Rationale:** To ensure each user has different roles and they do not have permission to access roles isn’t them. Therefore, the system need authorize for users based on account type (cashier, manager, staff) | | | |

### Element behavior

* + - 1. Add Store



* + - 1. Add Loyal Member



* + - 1. AddUser



* + - 1. AddUserComputer



* + - 1. Statistic



* + - 1. AddNewCategogy



* + - 1. AddNewProduct



* + - 1. Sale

Reference POS\_Sequence\_Sale.ipg

# Mapping between perspectives

**Legend:**

Begin with Data Model:

Begin with C&C View

Begin with Module View

# Tactics using for POS System

## Performance:

The system must handle requests quickly even when there are many requests

Introducing concurrency: Load balance can use to balance the requirements for the two server

Increase resources available: use two servers to be able to process multiple requests

Maintain multiple copies of either data or computations: When scanning a product code of the product, the system will get the product information and store it on a cache, in the second times scan that product, the product information will be retrieved from the cache

## Security:

Users want to log into the system will need to use one account and a password. Only accounts with the correct password can access the system.

* Authenticate users: Verify the system's users. Administrator will be created for an account and password for each user to be granted access to the system. Each different users will have different rights in the system, working with different data, this right will be confirmed when a new account
* Authorize users: Each account will have a different right in the system

When logging into the system, account information and passwords will be secured

* Maintain data confidentiality: using transmission Secure Sockets Layer (SSL) for a Web-based link to security information and account passwords

Information is transmitted needs to be integrity

* Maintain integrity: The information is transferred outside of the contained information is also part of checksum check to ensure the information is not lost on the transmission line

The account sales staff can only log into the system from the computer in sales system

Limit access: Use the computer's MAC address to be able to prevent the sign outside the system

# Reference:

<http://www.simple-talk.com/dotnet/.net-framework/.net-application-architecture-the-data-access-layer/>

<http://msdn.microsoft.com/en-us/library/dd381412(v=vs.98).aspx>

<http://www.asp.net/entity-framework>

<http://msdn.microsoft.com/en-us/library/bb399572(v=vs.90)>

<http://www.riskythinking.com/glossary/synchronous_replication.php>

<http://en.wikipedia.org/wiki/SOAP_(protocol)>

<http://newtech.about.com/od/newtechglossary/g/Odbc.htm>

<http://www.asp.net/mvc/tutorials/older-versions/controllers-and-routing/improving-performance-with-output-caching-cs>

<http://www.ehow.com/facts_5544243_benefits-client-server-network.html>

<http://en.kioskea.net/contents/cs/csintro.php3>