POS SYSTEM – ARCHITECTURE DRIVERS



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** |  |
| **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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Revision | Status | Change Summary | Revised by |
| 6/3/2012 | 1.0 |  | Consume Team member’s tasks | Thanh Giang |
| 15/4/202 | 1.1 |  | Update Quality Attributes | Giang Nguyen |
| 18/4/2012 | 1.2 |  | Usecase | Thanh Giang |
| 24/05/2012 |  | start | Use case Description | Hiep Ta |
| 9/6/2012 |  |  | Module View | Huy Huynh |

Contents

[**1.** **Document description** 4](#_Toc322214941)

[**1.1.** **Purpose and audience:** 4](#_Toc322214942)

[**1.2.** **Document organization:** 4](#_Toc322214943)

[**1.3.** **References and relevant document:** 6](#_Toc322214944)

[**2.** **Project overview** 6](#_Toc322214945)

[**2.1.** **Purpose** 6](#_Toc322214946)

[**2.2.** **Scope** 6](#_Toc322214947)

[**2.3.** **Stakeholders** 7](#_Toc322214948)

[**3.** **Architectural drivers** 8](#_Toc322214950)

[**4.** **System context** 8](#_Toc322214951)

[**5.** **Physic Perspective** 9](#_Toc322214952)

[**5.1.** **Primary presentation:** 9](#_Toc322214953)

[**5.2.** **Element catalog:** 9](#_Toc322214954)

[**5.2.1.** **Elements and their properties** 9](#_Toc322214955)

[**5.2.2.** **Relations and their properties** 10](#_Toc322214956)

[**5.2.3.** **Element interfaces** 11](#_Toc322214957)

[**5.2.4.** **Element behavior** 11](#_Toc322214958)

[**5.3.** **Context diagram** 11](#_Toc322214959)

[**5.4.** **Architecture background** 11](#_Toc322214960)

[**5.4.1.** **Rationale design** 11](#_Toc322214961)

[**5.4.2.** **Analysis of results** 12](#_Toc322214962)

[**5.4.3.** **Assumptions reflected in the design** 12](#_Toc322214963)

[**6.** **Static Perspective** 13](#_Toc322214964)

[**6.1.** **Primary presentation:** 13](#_Toc322214965)

[**6.2.** **Element catalog:** 14](#_Toc322214966)

[**6.2.1.** **Elements and their properties** 14](#_Toc322214967)

[**6.2.2.** **Relations and their properties** 16](#_Toc322214968)

[**6.3.** **Context diagram:** 16](#_Toc322214969)

[**6.4.** **Architecture background:** 16](#_Toc322214970)

[**6.5.** **Glossary of terms:** 17](#_Toc322214971)

[**6.6.** **Other information:** 17](#_Toc322214972)

[**7.** **Data Model** 17](#_Toc322214973)

[**8.** **Dynamic Perspective** 18](#_Toc322214974)

[**8.1.** **Primary presentation:** 18](#_Toc322214975)

[**8.2.** **Element catalog:** 19](#_Toc322214976)

[**8.2.1.** **Elements and their properties** 19](#_Toc322214977)

[**8.2.2.** **Relations and their properties** 20](#_Toc322214978)

[**8.2.3.** **Element behavior** 21](#_Toc322214979)

[**8.3.** **Context diagram:** 21](#_Toc322214980)

[**8.4.** **Architecture background:** 22](#_Toc322214981)

[**8.4.1.** **Rationale design** 22](#_Toc322214982)

[**8.4.2.** **Analysis of results** 22](#_Toc322214983)

[**8.4.3.** **Assumptions reflected in the design** 22](#_Toc322214984)

[**8.5.** **Glossary of terms:** 22](#_Toc322214985)

[**9.** **Solution background** 22](#_Toc322214986)

[**9.1.** **Architectural Approaches** 22](#_Toc322214987)

1. **Document description**
   1. **Purpose and audience:**

This document provides a high level overview of the evolving technical architecture for the Sales System of a retail chain using a loyalty card point system. It also provides a high-level description of the goals of the architecture, the use cases support by the system and architectural styles and components that have been selected to best achieve the use cases.

In addition to these views, this architectural description will:

* Identify the candidate patterns and tactics that will become the architecture design of the system.
* Frame the architectural design activity, begin with the given technical constraints and the structures
* Identify patterns generally suit the needs described in the product description based upon the quality attribute scenarios
* Identify tactics we apply to further refine the initial decomposition of the system and promote the necessary quality attributes.

The architecture has a set of guiding principles as well as known criteria and constraints that shape the proposed architecture. It is intended to capture and convey the significant architectural decisions which have been made on the system.

The development team can use this document to review the architecture of the system. The Architecture document will be also useful for future development teams.

* 1. **Document organization:**

Sub-sections of Section 1 include the following.

* Section 1.1: Purpose and audience: Describe who the intended audience and organizations are and what they might use the document for.
* Section 1.2: Document organization: Describe the overall organization of the document. List the major sections of the document and describe what concerns each section addresses.
* Section 1.3: Common notation: List any notation that will be used throughout the document.
* Section 1.4: Terminology and definitions: Define any terms used throughout the document and provide context for terminology.
* Section 1.5: References and relevant document: List any other relevant documents that the reader might need to refer to, and most importantly, describe their relationships to this document and why the reader might want to (or need to) refer to them.

Sub-sections of Section 2 include the following.

* Section 2.1 and 2.2: This section describes the project and its purpose and scope, why the system is being built.
* Section 2.3: List the relevant stakeholders, their organizations, and how they will interact with the system.

Sub-sections of Section 3 include the following: *In this section describe the architectural drivers for the system.*

* Section 3.1: Use-case diagram of the system and list the priority of the use-case
* Section 3.2: Business Constraints and Technical Constraint of the system and list the priority of these constraints
* Section 3.3: Quality Attributes and Technical Constraint of the system and list the priority of these Quality Attributes

Sub-sections of Section 4 include the following: The system context is the first step in design and should include at least one context drawing. In addition to the context drawing, we show the scope of the system being described by showing its relationship to external entities like systems, peripherals, organizations, and stakeholders as necessary to describe the context drawing. We also describe the relative perspective of the context drawing or drawings.

Sub-sections of Section 5-6-7-8-9 include the following: specify the software architecture. Views specify elements of software and the relationships between them. A view corresponds to a viewpoint and is a representation of one or more structures present in the software

Sub-sections of Section 10 include the following, which is an index of architectural elements and relations telling where each one is defined and used in this SAD. The section also includes a glossary and acronym list.

* 1. **References and relevant document:**

|  |  |
| --- | --- |
| **Name** | **Description** |
| **Sales System of a retail chain using a loyalty card point system** | System description |
| **Architecting software intensive systems** | A practitioner’s guide, Anthony J. Lattanze |

1. **Project overview**
   1. **Purpose** **and Scope**

The project will aim to develop a sale system for Company A, a retail chain (hereinafter, the system) in conjunction with its launch of a point service.

The system consists of a main server, located at the head office. The head office server and the POS terminals are connected to each other via a network. Products sold at stores have bar codes attached which indicate the product codes. These bar codes can be read with bar code readers of POS terminals. Customer who have become point service members are issued point cards, which bear bar codes indicating their member numbers, and when they purchase products with cash, they are awarded points based on the amount of their purchase.

For each product, its standard price, common to all stores, is set as a part of the product data. Each store, however, can set and use its own actual retail price instead of the standard price during the limited period specified is each store. The actual retail price must be set in advance, and it cannot be charged in the middle of the specified period.

Products are classified into product types such as food, general merchandise, etc. Not all stores carry every product type, and the range of product types carried is designated for each store.

Customer also use online website at everywhere to check their information such as personal information, loyal point.

* 1. **Stakeholders**

|  |  |  |
| --- | --- | --- |
| **Stakeholders** | **Organizations** | **Interaction** |
| **Customers** | None | Can check their point at online store website |
| **Staffs** | Company A | Responsible for manage information of products, categories, customer, |
| **Cashiers** | Company A | Responsible for check bills (records) |
| **Manager** |  | Responsible for manage retail stores, and they can statistic sales by many criterion |
| **Administrator** | Company A | Responsible for manage user of the system such as: Create new, assign authorize. He can also manage computer of user computer so they can access while they are not working |

1. **Architectural drivers**

***Please reference to file “Architectural Driver”***

1. **System context**



**Users and roles:** Stakeholders who interact to Retail system was described in section 2.3.Stakeholder of this document.

**Channels:** Users will use different channels to access the system.

* Staff: Use website on PC locates at Retail Store to access to system.
* Manager: use website on everywhere PC to access to system.
* Cashier: Use Bar code reader at computer using store 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

1. **Physic Perspective**

**ALLOCATION VIEW**

**(Deployment Style)**

* 1. **Primary presentation:**



* 1. **Element catalog:**
     1. **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 Service | * It’s a WEB application | * See detail at Section 5.3 |
| 5 | Database Server | * OS: Windows Server 2008 * Processor: 1 x Intel® Xeon® Processor E5606 (8M Cache, 2.13 GHz, 4.80 GT/s Intel® QPI) * Memory: 1 x 2GB DDR3 1333 240-Pin DDR3 ECC RDIMMs/UDIMMs (PC3 10666) * Hard Disk: DELL 250GB SATA 7.2K 3.0Gbs 3.5" Enterprise * MainBoard: Asus Serverboard Z8NA-D6 * Chassis : USP 100-450 * RAID Configuration: Intel® ICH10R:6 x SATA2 300MB/s * Intel® Matrix Storage (For Windows Only) * Soft: Microsoft SQL Server 2008 Enterprise | * Run Main Database * Run Backup Database |
| 6 | User PC | * Operation System : Genuine Windows® 7 Home Basic, * Processor: AMD AM3 For Phenom™ II/Athlon™ II Family /Processors * Chipset: AMD SB710 * Memory: 2 x DIMM Slots Dual Channel DDR3 1066/1333 MHz, Support max 4 GB. * Graphics: ATI Radeon HD 3200, Support Micorsoft DirectX 10, Choices: NVIDIA GeForce G210 512MB * Hard Drive SATA: 3.5" 320G * Card Reader: MS/MS Pro/MMC/SD * LAN/WLAN: LAN: 10/100/1000 * Audio: Azalia 8 Channel | * Head office   + Run Head Office Application * Store   + Run Store Application   + Run Store Database |
| 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 |

* + 1. **Relations and their properties**
    2. **Element behavior**
  1. **Architecture background**
     1. **Rationale design**
* Deployment design satisfy quality attributes following:
  + Available: each store is installed a Store Database to ensure that when Head Office Database is disconnect system still operate in normal.
  + Performance: main sale data is stored at store, so request speed from Store Application to Store Database increase.
  + Scalability: stores is designed in dependently, so easy to add or remove a store on system.
    1. **Analysis of results**
    2. **Assumptions reflected in the design**

1. **Static Perspective**

**Module View**

**(Layered Style and Uses Style)**

* 1. **Primary presentation:** 
     1. **MVC Pattern**



* + 1. **MVC Uses Style**



* + 1. **Layereds Style**



* 1. **Element catalog:**
     1. **Elements and their properties**
        1. **MVC**

|  |  |  |
| --- | --- | --- |
| Elements | | Properties |
| Controller | **ProductController** | The ProductController class contains action methods that render view pages (AddProduct, EditProduct, ViewProduct) |
| **Category**  **Controller** | The CategoryController class contains action methods that render view pages (AddCategory, EditCategory, ViewCategory) |
| **StatisticsController** | The StatisticsController class contains action methods that render view pages (Statistics) |
| **SaleController** | The SaleController class contains action methods that render view pages (BillManagement, Checkout, PriceLog) |
| **StoreController** | The StoreController class contains action methods that render view pages (AddStore, ViewStore, EditStore) |
| **LoyalMember**  **Controller** | The LoyalMemberController class contains action methods that render view pages (AddLoyalMember, ViewLoyalMember, EditLoyalMember) |
| **StoreCategory**  **Controller** | The StoreCategoryController class contains action methods that render view pages (AddStoreCategory, ViewStoreCategory, EditStoreCategory) |
| **POSController** | The POSController class contains action methods that render view pages (AddPOS, ViewPOS, EditPOS) |
| **UserController** | 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. |
| **PriceLog** | This GUI shows all Price history that was used for each product |
| **Checkout** | This GUI helps the user check bill and make a payment. |
| **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. |
| **Statistics** | To make statistics about the total amount of product (or product category) was bought on month. |
| Model | **ProductModel** | Storing and retrieving the Product information and return a message back to view pages (AddProduct, EditProduct, ViewProduct) |
| **SaleModel** | Storing and retrieving the Sale information and return a message back to view pages (BillManagement, Checkout, PriceLog) |
| **StoreModel** | Storing and retrieving the Retail Store information and return a message back to view pages (AddStore, EditStore, ViewStore) |
| **LoyalMember**  **Model** | Storing and retrieving the Customer information and return a message back to view pages (AddLoyalMember, EditLoyalMember, ViewLoyalMember) |
| **UserModel** | Storing and retrieving the Userinformation and return a message back to view pages (AddUser, EditUser, ViewUser) |
| **CategoryModel** | Storing and retrieving the Product Categoryinformation and return a message back to view pages (AddProductCategory, EditProductCategory, ViewProductCategory) |
| **StoreCategory**  **Model** | Storing and retrieving the Store Category information and return a message back to view pages (AddStoreCategory, EditStoreCategory, ViewStoreCategory) |
| **POSModel** | Storing and retrieving the POSinformation and return a message back to view pages (AddPOS, EditPOS, ViewPOS) |
| **StatisticsModel** | Storing and retrieving the some information related Statistics and return a message back to view pages (Statistics) |

* + - 1. **Layer Style**

|  |  |  |
| --- | --- | --- |
| Elements | | Properties |
| Data Access Layer | **ProductDA** | Consists of function to retrieve data from database or store, update data - which related Product - to database. It is frequently called by class AddProduct, EditProduct, ViewProduct, Checkout, PriceLog. |
| **BillDA** | Consists of function to retrieve data from database or store, update data - which related Bill - to database. It is frequently called by class BillManagement, Statistics, Checkout. |
| **RetailStoreDA** | Consists of function to retrieve data from database or store, update data - which related RetailStore - to database. It is frequently called by class EditStore, AddStore, ViewStore. |
| **CustomerDA** | Consists of function to retrieve data from database or store, update data - which related Customer - to database. It is frequently called by class AddLoyalMember, EditLoyalMember, ViewLoyalMember, Checkout. |
| **UserDA** | Consists of function to retrieve data from database or store, update data - which related UserAccount - to database. It is frequently called by class AddUser, EditUser, ViewUser. |
| **BillDetailDA** | Consists of function to retrieve data from database or store, update data - which related BillDetail - to database. It is frequently called by class Statistics, BillManagement, Checkout. |
| **CategoryDA** | Consists of function to retrieve data from database or store, update data - which related Category - to database. It is frequently called by class AddCategory, ViewCategory, EditCategory, ViewProduct, EditProduct, AddProduct, Checkout, PriceLog. |
| **RetailStoreCategory**  **DA** | Consists of function to retrieve data from database or store, update data - which related RetailStoreCategory - to database. It is frequently called by class AddStoreCategory, ViewStoreCategory, EditStoreCategory, AddStore, ViewStore, EditStore. |
| **POSDA** | Consists of function to retrieve data from database or store, update data - which related POS - to database. It is frequently called by class AddPOS, ViewPOS, EditPOS. |
| **CostDA** | Consists of function to retrieve data from database or store, update data - which related Cost - to database. It is frequently called by class PriceLog. |
| **Context** | 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. |

* 1. **Architecture background:**

System was separate into three layers include: Presentation layer, Transfer Data Object Layer, Data Access Layer.

The reason why we use three layers instead of four layers (such as: Presentation layer, Business Logic layer, Data Object Transfer Layer, Data Access Layer) is:

* The controller in MVC undertake tasks that is implemented on Business Logic Layer.
* On Web environment, increasing performance is important. Therefore, eliminating a layer is needed.

Let’s get to the bottom of three layers:

* Presentation Layer (Including MVC framework): is responsible for communication with end user to collect data and show a result of data through components in user interface. We combine MVC framework with the Presentation Layer. For the benefit of MVC framework, we will talk about this later, after show all features of three layers.
  + 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, perform 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.
* Data Transfer Object: this layer can creates objects to support Presentation Layer to perform and used is as a transfer object and pass-by-reference to Data Access Layer.
* Data Access Layer manages the physical storage and retrieval of data from database.

Besides using 3 layers, we also use MVC framework. 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.
* *Multiple GUI platforms supported.* MVC allows targeting different GUI platforms such as: Window, Web, Silverlight, etc,… Therefore, the same application can be used with quite different presentation layers - one for Windows, the other for Silverlight or Web environment, etc.:



### Platform-independent navigation to views. To make application logic fully independent of the presentation layer, MVC provides a platform-independent way of navigating to views. Instead of  activating a Windows form or redirecting to a Web page a developer just simply call a uniform Navigator.Navigate(...) method. For example:

public class OrderDetailsController

...

public void ProcessOrder()

{

// No Response.Redirect(...) or Form.Show() calls

Task.Navigator.Navigate(OrderSupportTask.ProcessOrder);

}

### Tasks concept. Sometime, we have to unites 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.

### 

* 1. **Glossary of terms:**
  2. **Other information:**

1. **Data Model**



|  |  |  |  |
| --- | --- | --- | --- |
| Entity | Attributes | Data Type | Description |
| Bill | **Bill\_ID** | VARCHAR(11) |  |
| **POS\_ID** | VARCHAR(11) | Attribute said bill is made in which POS well as the general store |
| **Customer\_ID** | VARCHAR(9) | Customer pays the invoice. |
| **User\_ID** | VARCHAR(9) | Cashier |
| TotalCost | INT | 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 | **Product\_ID** | VARCHAR(11) |  |
| Product\_Name | NVARCHAR(50) |  |
| BasicCost | INT | AttributeBasic Cost show the default price of the product |
| **Category\_ID** | VARCHAR(11) |  |
| Bill\_Detail | **Bill\_ID** | VARCHAR(11) |  |
| **Product\_ID** | VARCHAR(11) |  |
| Quantity | INT | Sum Loyal Point of customer |
| Customer | **Customer\_ID** | VARCHAR(9) |  |
| Customer\_Name | NVARCHAR(50) |  |
| Customer\_Address | NVARCHAR(50) |  |
| Customer\_Phone | INT |  |
| SumPoint | INT | Sum Loyal Point of customer |
| RetailStore | **RetailStore\_ID** | VARCHAR(11) |  |
| RetailStore\_Name | NVARCHAR(50) |  |
| Cost | **Product\_ID** | VARCHAR(11) | Entity Cost said that Retail Store Retail Store selling a certain product and pricing individual products within a certain time. |
| **RetailStore\_ID** | VARCHAR(11) |
| DateStart | DATETIME |
| DateEnd | DATETIME |
| Cost | INT |
| Category | **Category\_ID** | VARCHAR(11) |  |
| Category\_Name | NVARCHAR(50) |  |
| RetailStore\_Category | **RetailStore\_ID** | VARCHAR(11) |  |
| **Category\_ID** | VARCHAR(11) |  |
|  | Quantity | INT |  |
| Computer | **Computer\_MAC** | VARCHAR(11) |  |
| **RetailStore\_ID** | VARCHAR(11) | This Attribute tells us this POS Terminal is placed at which Retail Store |
| User | **User\_ID** | VARCHAR(9) |  |
| User\_Name | NVARCHAR(50) |  |
| User\_Address | NVARCHAR(50) |  |
| User\_Phone | INT |  |
| Password | INT |  |
| **RetailStore\_ID** | VARCHAR(11) | This Attribute tells us this user works at which Retail Store |

1. **Dynamic Perspective**

**Component and Connector View**

* 1. **Primary presentation:**

**Head Office C&C View**



**Store C&C View**



**Synchronic Database**



* 1. **Element catalog:**
     1. **Elements and their properties**

|  |  |  |
| --- | --- | --- |
| **Elements** | | **Properties** |
| **Database Server** | **Head Office DB Server** | Database server which locates at Head Office is responsible for store data such as sales data, user data, customer data, store data, product data, and category data. This is where Head Office PC gets data to perform statistical analysis. |
| **Client DB Server** | Database server which locates at POS terminal and responsible for store data of stores such as product cost, user information and bill detail. It also is a reserved database server, store as much as possible data when Head Office server is going down or connect problem happen. |
| **User Interface** | **Cashier Interface** | This interface use for cashier to perform sales activities and allow cashier interact with product and loyal point information. |
| **Administrator Interface** | This interface use for administrator to perform system operating action. It allow administrator have authorities at user account and synchronize data. |
| **Staff Interface** | This interface use for staff to manages information about customer category, product. It also allows staff gets data from system and performs statistical analysis. |
| **Object** | **Loyal Point** | This function allow user view customer loyal point. |
| **Sale** | All function relate to sale activities which perform by cashier |
| **Category** | Contain functions such as view, add, update and remove category supports staff performs manage activities. |
| **Product** | Contain functions such as view, add, update and remove product supports staff performs manage activities. |
| **Synchronize data** | These functions contain set time for auto synchronize activities or manually synchronize |
| **Statistical analysis** | Use by staff to collect sales data and generates analysis for demand |
| **User account** | Use by administrator, contain add, update information of system users |
| **Customer** | Use by administrator, contain add, update information of system customers. |

* + 1. **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 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 |
| **Replication** |  |

* + 1. **Element behavior**
       1. Head Office Behavior

Add POS



Statistical Analysis



Add new product



* + - 1. Store Behavior

Sale product:

With customer has point card



With normal customer



Scan product



* 1. **Context diagram:**



* 1. **Architecture background:**
     1. **Rationale design**

The system includes 2 Database Server located in Head Office and Store and application uses data from these servers for sales activities as well as data storage. So in this architecture will use the Call-Return Styles, include Client-Server style and a Call-return style that objects call other objects referred to other data and wait for the return data from them. The called object will be call data from repository use Client-server styles. They will send request to database server and wait for the reply from them.

* + 1. **Analysis of results**

When use Client-Server style with 2 database server located at 2 different places we can improve system availabilities

* + 1. **Assumptions reflected in the design**

There will have reserve database server in store, which is responsible for storing product information daily and sales information to sync up to Head Office server and performing the redundancy while Head Office server going down.

* 1. **Glossary of terms:**

**DB Server:** Database Server is a computer program that provides database services to other computer programs or computers.

**POS Terminal:** A point-of-sale terminal is a computerized replacement for a cash register.

1. **Solution background**
   1. **Architectural Approaches**
      1. **Using client-server style in LAN.**

The main benefits of the client/server architectural style are:

* **Higher security**. All data is stored on the server, which generally offers a greater control of security than client machines.
* **Centralized data access**. Because data is stored only on the server, access and updates to the data are far easier to administer than in other architectural styles.
* **Ease of maintenance**. Roles and responsibilities of a computing system are distributed among several servers that are known to each other through a network. This ensures that a client remains unaware and unaffected by a server repair, upgrade, or relocation.
* **Ease to add more client into the system** 
  + 1. **Ensuring availability:**

When server isn’t available, computers in POS can continue to perform a progress that doesn’t require time-out.

* Passive redundancy: Allocate subsidiary database in POS, when the transactions are performed, which will store data into database in POS and head office. When database in Head office isn’t available, data still keep.

When incident is occurred to a payment in POS, the progress was being performed before that will be canceling.

* Transactions: When the incident is occurred in any step of payment process, all of transaction will be canceling!

* + 1. **Ensuring Performance:**

**Ensuring operation in POS that should be performed rapidly.**

* Reduce computational overhead: To keep it is quickly and timely in performing operation, the system will reduce a communication between tiers with each other. However, the mobility of system will be decreased.
* Increase available resources: increase hardware (real-time processing, memory, transmission...). However, the cost will be increased.

1. **Mapping between perspectives** 
   1. **Mapping between a module view and a component-and-connector view**

**Head Office mapping:**



**Store Mapping**



* 1. **Mapping between module view and allocation view.**



|  |  |
| --- | --- |
| **Element in Allocation View** | **Element in Module View (Data Model)** |
| **Main Database** | **Head Office:**   * + Bill   + Bill\_Detail   + User   + Product   + Category   + Cost   + POS   + Customer   + RetailStore\_Category |
| **Local Database** | **Retail Store:**  • Bill  • Bill\_Detail  • User\_RT  • Product\_RT  • Category\_RT |

**Document directory**

In this section include an index, glossary, and acronym list

|  |  |
| --- | --- |
|  |  |

## -- The End --