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 |

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 |

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 head office server, located at the head office, and the POS terminals placed at store cashiers. 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.

The system was built to connect retail stores and Head Office which located in the same building through Local Area Network. The Head Office and the POS terminals are connect with Database servers, Head Office are connect with two database server (One run, one reserve), the reserve database server use tactic active redundancy. Both database servers at Head Office and POS terminal can be accessed during the sales operation.

Moreover, in addition to the sales operation, the data synchronization between the store and the Head Office will be divided into two types. One is manual-synchronized by the administrator in the beginning and at the end of day. This type of synchronization will sync all the information between Head Office database and POS terminal database. The other is auto-synchronized by the system every 5 minutes for the sale and customer information such as bills, loyal point to the system, so the system can perform statistical analysis on the sales records of all stores in near real-time manner.

Customer also use touch screen at store to check their information such as personal information, loyal point.

* 1. **Stakeholders**

***Sẽ bổ sung sau***

|  |  |  |
| --- | --- | --- |
| **Stakeholders** | **Organizations** | **Interaction** |
| **Customers** | None |  |
| **Staffs** | Company A | Responsible for manage information of products, categories, customer, retail stores, POS and they can statistic sales by many criterion |
| **Cashiers** | Company A | Responsible for check bills (records) |
| **Administrator** | Company A | Responsible for manage user of the system such as: Create new, assign authorize. He can also sync information between |

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 PC locates at Head Office to access to system.
* Cashier: Use Bar code reader at POS terminal to interacts with system
* Customer: Use touch screen at POS terminal to access the system
* 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**

**At Head Office:**

1. There is a primary server which will access and request the access for the computer in the whole system and store the main database.
2. There is a reserve server which is used when the primary server is corrupt (is not connected, Database is broken, the software is damaged). This machine also runs parallel tasks with primary Server to perform hosting the database (in case the worst happens is that the damage, the Database Server is stable and can make mistakes at least).
3. At the control center will have multiple computers installed manger application for the center.
4. The system will have 02 main Firewall. Firewall security and control all connections (request, access ...) from within the store or the control center.

**At Retail Store:**

1. There is a primary Server which will access and request the access for the computer and used to store the entire database of the store.
2. At the store will have the cash which installed computer programs for the control store Sales Application for the store will only for cashier
3. There also need to configure the firewall as in Head Office.
   * 1. **Relations and their properties**

**At Head Office:**

* The computers are connected together through an Ethernet switch.
* From this switch, the computer will be connected directly to the main Server.

**At Retail Store:**

* The cashier’s computer is connected through an Ethernet switch.
* This Switch also has the task of connecting to a LAN which has Retail Stores and Head Office
  + 1. **Element interfaces**

**At Head Office:**

* The computers will be installed at the Head Office GUI
* Configuration of the machine is: Windows 7 - IBM 235

**At Retail Store:**

* The computers will be installed at the GUI of Retail
* Configuration of the machine is: Windows 7 - IBM 235
  + 1. **Element behavior**

***(Liên hệ với các view còn lại)***

* 1. **Context diagram**



* 1. **Architecture background**
     1. **Rationale design**
* At the Head Office have 2 Server running in parallel to meet the availability attribute of data when the primary Server has technical problems and cannot perform the tasks.
* Firewall will help avoid unwanted connection to the Server.
* Switch will connect computers together into a LAN. And connect the computers with the Server.
* Each Retail Store has a separate server to meet the fastest database access. Improving performance of the entire QA system and through which also ensures QA Availability
  + 1. **Analysis of results**

The error event occurs, the system will be handled:

**Head Office:**

* In case of damage, the Reserve Server will take the place of the main Server
* As such, the entire performance of the overall QA system is a normal response.
* Availability and QA still be assured of the highest status.

**Retail Store:**

* The Retail Store disconnected with the system (control center) for one reason or another (not connected to the LAN, the software does not work ...), customers will not be paid by the loyalty points
  + 1. **Assumptions reflected in the design**
* This network connection between Head Office and Retail Store may be damaged
* The main Server at Head Office can be damaged and not work
* One of the computers at the Head Office may be damaged and cannot connect to the system
* One of the computers in Retail also damaged and cannot connect to the system

1. **Static Perspective**

**Module View**

**(Decomposition Style and Layer Style)**

* 1. **Primary presentation:** 
     1. **Head Office**



* + 1. **Retail Store**



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

|  |  |  |
| --- | --- | --- |
| **Elements** | | **Properties** |
| **GUI Layer** | **POSManageUI** | This interface is necessary for the manager to monitor Post Offices System (POS) in store. The manager can add new POS, modify POS information or delete POS. |
| **UserAccountUI** | Helps the admin manages user accounts such as: add new user, modify user account information, active or deactivate user. |
| **LoyalMemberUI** | Manages members who bought products at store frequently and their loyal point. |
| **StoreCategoryUI** | The manager can monitor category of stores that mean the group of stores have some characteristics is similar. The manager can add new category, modify category information or delete category. Moreover, the manager also adds some retail store to store category or removes it from category. |
| **ProductHOUI** | Helps the manager monitor products. The manager can add new product, modify the product information or delete the product. |
| **CostHistoryUI** | Thanks to this interface, the manager can know the price of product after several changes. |
| **CategoryUI** | The manager can monitor category of product that mean the group of product have some characteristics is similar. The manager can add new category, modify category information or delete category. Moreover, the manager also adds some product to store category or remove it from category. |
| **StatisticUI** | This interface will help manager to make a statistics about revenue or best-selling item in month or year. |
| **StoreManageUI** | Thank to this interface, the manager monitor store such as: add new store, modify store information or delete store. |
| **SettingUI** | This interface will help you setting the system such as:   * Setting synchronous data includes two modes: manual synch and automatic sync.   + Manual sync: When the user choose this, sync data is manually operated.   + Automatic sync: The user set time for sync data. After a period of time user set, the data will be synchronized. * The interface includes setting loyal point for each product. |
| **Business Logic Layer** | **LoyalMemberBL** | This class includes functions related about Loyal Member. It support for **LoyalMemberUI** class which uses function of **LoyalMemberBL**.  The functions in **LoyalMemberBL** will handle events that were executed in **LoyalMemberUI** and return a result. Sometime, **LoyalMemberBL** create some objects from some classes of DTO layer as a parameter. **LoyalMemberBL** uses this parameter when call some functions in Data Access Layer as a reference and the return of these functions will return to interface. |
| **POSManageBL** | This class includes functions related about Loyal Member. It support for **POSManageUI** class which uses function of **POSManageBL**.  The functions in **POSManageBL** will handle events that were executed in **POSManageUI** and return a result. Sometime, **POSManageBL** create some objects from some classes of DTO layer as a parameter. **POSManageBL** uses this parameter when call some functions in Data Access Layer as a reference and the return of these functions will return to interface. |
| **StatisticBL** | This class includes functions related about Loyal Member. It support for **StatisticUI** class which uses function of **StatisticBL**.  The functions in **StatisticBL** will handle events that were executed in **StatisticUI** and return a result. Sometime, **StatisticBL** create some objects from some classes of DTO layer as a parameter. **StatisticBL** uses this parameter when call some functions in Data Access Layer as a reference and the return of these functions will return to interface. |
| **ProcessHOBL** | This class manages all progress of the system in Head Office. Monitoring objects of other classes (includes GUI layer and Business Logic Layer) and some constant variables. |
| **CostHistoryBL** | This class includes functions related about Loyal Member. It support for **CostHistoryUI** class which uses function of **CostHistoryBL**.  The functions in **CostHistoryBL** will handle events that were executed in **CostHistoryUI** and return a result. Sometime, **CostHistoryBL** create some objects from some classes of DTO layer as a parameter. **CostHistoryBL** uses this parameter when call some functions in Data Access Layer as a reference and the return of these functions will return to interface. |
| **SettingBL** | When user setting something, it will call a function in SettingBL to handle it and store some settings in XML file. When the system starts to operate the next-time, the system will use settings in XML file. |
| **CategoryHOBL** | This class includes functions related about Loyal Member. It support for **CategoryUI** class which uses function of **CategoryHOBL**.  The functions in **CategoryHOBL** will handle events that were executed in **CategoryUI** and return a result. Sometime, **CategoryHOBL** create some objects from some classes of DTO layer as a parameter. **CategoryHOBL** uses this parameter when call some functions in Data Access Layer as a reference and the return of these functions will return to interface. |
| **CommonHOBL** | Includes functions that all of classes in Business Logic Layer can use of Head Office. |
| **UserHOBL** | This class includes functions related about Loyal Member. It support for **UserAccountUI** class which uses function of **UserHOBL**.  The functions in **UserHOBL** will handle events that were executed in **UserAccountUI** and return a result. Sometime, **UserHOBL** create some objects from some classes of DTO layer as a parameter. **UserHOBL** uses this parameter when call some functions in Data Access Layer as a reference and the return of these functions will return to interface. |
| **ProductHOBL** | This class includes functions related about Loyal Member. It support for **ProductHOUI** class which uses function of **ProductHOBL**.  The functions in **ProductHOBL** will handle events that were executed in **ProductHOUI** and return a result. Sometime, **ProductHOBL** create some objects from some classes of DTO layer as a parameter. **ProductHOBL** uses this parameter when call some functions in Data Access Layer as a reference and the return of these functions will return to interface. |
| **StoreManageBL** | This class includes functions related about Loyal Member. It support for **StoreManageUI** class which uses function of **StoreManageBL**.  The functions in **StoreManageBL** will handle events that were executed in **StoreManageUI** and return a result. Sometime, **StoreManageBL** create some objects from some classes of DTO layer as a parameter. **StoreManageBL** uses this parameter when call some functions in Data Access Layer as a reference and the return of these functions will return to interface. |
| **StoreCategoryBL** | This class includes functions related about Loyal Member. It support for **StoreCategoryUI** class which uses function of **StoreCategoryBL**.  The functions in **StoreCategoryBL** will handle events that were executed in **StoreCategoryUI** and return a result. Sometime, **StoreCategoryBL** create some objects from some classes of DTO layer as a parameter. **StoreCategoryBL** uses this parameter when call some functions in Data Access Layer as a reference and the return of these functions will return to interface. |
| **SyncBL** | Contain constant variables fixed the sync data mode such as: Manual mode or automatic mode and the specific time to sync. |
| **Data Transfer Object** | **ProductHODTO** | Includes variables only belong to ProductHODTO (Examples: ProductID, ProductName, BasicCost, CategoryID), not include functions. |
| **BillHODTO** | Includes variables only belong to BillHODTO (Examples: BillID, RetailStoreID, CustomerID, UserID, Date, TotalPoint, PlusPoint, MinusPoint), not include functions. |
| **UserHODTO** | Includes variables only belong to UserHODTO (Examples: UserID, UserAddresss, UserName, UserPhone, Password, RetailStoreID), not include functions. |
| **POSDTO** | Includes variables only belong to POSDTO (Examples: POSID, POSName, RetailStoreID), not include functions. |
| **CostHistoryDTO** | Includes variables only belong to CostHistoryDTO (Examples: RetailStoreID , ProductID, DateStart, DateEnd, Cost), not include functions. |
| **StoreDTO** | Includes variables only belong to StoreDTO (Examples: RetailStoreID , RetailStoreName), not include functions. |
| **BillDetailHODTO** | Includes variables only belong to BillDetailHODTO (Examples: BillID, ProductID, Quantity), not include functions. |
| **CategoryHODTO** | Includes variables only belong to CategoryHODTO (Examples: CategoryID, CategoryName, Quantity), not include functions. |
| **LoyalMemberDTO** | Includes variables only belong to LoyalMemberDTO (Examples: MemberID, MemberName, MemberAddress, MemberPhone, Sumpoint), not include functions. |
| **StoreCategoryDTO** | Includes variables only belong to StoreCategoryDTO (Examples: StoreCategoryID, StoreCategoryName), not include functions. |
| **Data Access Layer** | **BillHODA** | Includes function to retrieve data from database or store, update data - which related Bill - to database. It is frequently called by class StatisticBL. |
| **UserHODA** | Includes function to retrieve data from database or store, update data - which related User Account - to database. It is frequently called by class UserHOBL. |
| **POSHODA** | Includes function to retrieve data from database or store, update data - which related POS - to database. It is frequently called by class UserHOBL. |
| **BillDetailHODA** | Includes function to retrieve data from database or store, update data - which related Bill Detail - to database. It is frequently called by class StatisticBL. |
| **CategoryHODA** | Includes function to retrieve data from database or store, update data - which related Category - to database. It is frequently called by class CategoryHOBL. |
| **MemberDA** | Includes function to retrieve data from database or store, update data - which related Loyal Member - to database. It is frequently called by class LoyalMemberBL. |
| **StoreCategoryDA** | Includes function to retrieve data from database or store, update data - which related Store Category - to database. It is frequently called by class StoreCategoryBL. |
| **ProductHODA** | Includes function to retrieve data from database or store, update data - which related Product - to database. It is frequently called by class ProductHOBL. |
| **StoreDA** | Includes function to retrieve data from database or store, update data - which related Store - to database. It is frequently called by class StoreManageBL. |
| **CostHistoryDA** | Includes function to retrieve data from database or store, update data - which related Cost History - to database. It is frequently called by class CostHistoryBL. |

* + - 1. **Retail Store**

|  |  |  |
| --- | --- | --- |
| **Elements** | | **Properties** |
| **GUI Layer** | **ProductRTUI** | Helps the cashier monitor products at the store. |
| **SaleUI** | This interface supports cashier to pay a bill. The casher fills a code or ID of product and performs a payment. It includes pay by loyal point or in cash. It can call some calculation function in SaleBL to perform a bill. |
| **Business Logic Layer** | **CommonRTBL** | Includes functions that all of classes in Business Logic Layer can use of Retail Store. |
| **UserRTBL** | Perform user account information to login into system. |
| **ProcessRTBL** | This class manages all progress of the system in Retail Store. Monitoring objects of other classes (includes GUI layer and Business Logic Layer) and some constant variables. |
| **SaleBL** | This class includes functions related about Loyal Member. It support for **SaleUI** class which uses function of **SaleBL**.  The functions in **SaleBL** will handle events that were executed in **SaleUI** and return a result. Sometime, **SaleBL** create some objects from some classes of DTO layer as a parameter. **SaleBL** uses this parameter when call some functions in Data Access Layer as a reference and the return of these functions will return to interface. |
| **LoyalPointBL** | This class includes functions related about Loyal Member. It support for **LoyalPointUI** class which uses function of **LoyalPointBL**.  The functions in **LoyalPointBL** will handle events that were executed in **LoyalPointUI** and return a result. Sometime, **LoyalPointBL** create some objects from some classes of DTO layer as a parameter. **LoyalPointBL** uses this parameter when call some functions in Data Access Layer as a reference and the return of these functions will return to interface. |
| **ProductRTBL** | This class includes functions related about Loyal Member. It support for **ProductRTUI** class which uses function of **ProductRTBL**.  The functions in **LoyalPointBL** will handle events that were executed in **ProductRTUI** and return a result. Sometime, **ProductRTBL** create some objects from some classes of DTO layer as a parameter. **ProductRTBL** uses this parameter when call some functions in Data Access Layer as a reference and the return of these functions will return to interface. |
| **Data Transfer Object** | **ProductRTDTO** | Includes variables only belong to ProductRTDTO (Examples: ProductID, ProductName, BasicCost, CategoryID), not include functions. |
| **BillRTDTO** | Includes variables only belong to BillRTDTO (Examples: BillID, RetailStoreID, CustomerID, UserID, Date, TotalPoint, PlusPoint, MinusPoint), not include functions. |
| **UserRTDTO** | Includes variables only belong to UserRTDTO (Examples: UserID, UserName, Password), not include functions. |
| **BillDetailRTDTO** | Includes variables only belong to BillDetailRTDTO (Examples: BillID, ProductID, Quantity), not include functions. |
| **Data Access Layer** | **BillRTDA** | Includes function to retrieve data from database or store, update data - which related Bill - to database. It is frequently called by class SaleBL. |
| **UserRTDA** | Includes function to retrieve data from database or store, update data - which related Product - to database. It is frequently called by class UserRTBL. |
| **ProductRTDA** | Includes function to retrieve data from database or store, update data - which related Product - to database. It is frequently called by class ProductRTBL. |
| **BillDetailRTDA** | Includes function to retrieve data from database or store, update data - which related Bill Detail - to database. It is frequently called by class SaleBL. |

* + 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. |

* 1. **Context diagram:**



* 1. **Architecture background:**

System was separate into 4 layers include: GUI layer, Business Logic Layer, Transfer Data Object Layer, Data Access Layer.

When developing 4-tier, it will reduce maintenance and development time, because of separate following to N-tier, it looks like divide and conquer method. When we change some functions of this layer, then, it doesn’t affect much to other layers, so, maintenance cost is lower; easier to upgrade or modify. Each layer can be reuse anytime in other applications.

The layer can be run on different OS from each other. Example: Database can be run on UNIX or LINUX; and GUI layer can be run on Windows (or Web server) and are developed by many programming language and team development.

Present layers and its purpose:

* GUI layer: is responsible for communication with end user to collect data and show a result of data through components in user interface. This layer can use some service that Business Logic layer provide. According the requirement, the GUI layer includes: ProductUI, CustomerUI, CategoryUI, StatisticUI, ManagementUI, and SaleUI.
* Business Logic Layer: perform data to display on UI or store to database. Business logic layer can check and do following operation required, the capabilities of program will execute logictic. This layer includes: ProductBL, CustomerBL, CategoryBL, SaleRetailStoreBL, StatisticBL, UserAccountBL, and Common.
* Data Access Layer: this layer is responsible to perform requirements of Business Logic Layer to database such as: update database, read data from database and return to Business Logic Layer. This layer includes: UserDB, ProductDB, CustomerDB, CategoryDB, and SaleRetailStoreDB.
* Data Transfer Object: this layer can creates objects to support Business Logic Layer to perform and used in transfer parameter to Data Access. This layer include: ProductDTO, CategoryDTO, CustomerDTO, SaleRetailStoreDTO, and UserDTO.
  1. **Glossary of terms:**
  2. **Other information:**

1. **Data Model**



|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Attributes** | **Data Type** | **Description** |
| **Head Office** | | | |
| **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 |  |
| **POS** | **POS\_ID** | VARCHAR(11) |  |
| POS\_Name | NVARCHAR(50) |  |
| **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 |
| **Retail Store** | | | |
| **Bill** | **Bill\_ID** | VARCHAR(11) | Bill Information will be synchronized regularly every 5 minutes, get ready for performing Staff report statistics accurately. |
| **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. Then once every five minutes will be synchronized with the Head Office to work out SumPoint for Customer entity.  If, when disconnected, the Customer cannot perform the function used to pay points (MinusPoint will not save), but the plus point of the bill that will still be written in this field. When the system back to normal state, in PlusPoint data will be synchronized. |
| MinusPoint | INT |
| **Bill\_Detail** | **Bill\_ID** | VARCHAR(11) |  |
| **Product ID** | VARCHAR(11) |  |
| Quantity | INT | Quantity of a product |
| **User\_RT** | **User\_ID** | VARCHAR(9) |  |
| User\_Name | NVARCHAR(50) |  |
| Password |  |  |
| **Product\_RT** | **Product\_ID** | VARCHAR(11) | Manage products that stores that sell. Product information will be updated in the early days from Head Office to the Store |
| Product\_Name | NVARCHAR(50) |  |
| Cost | INT | -This Attribute will be updated from the Cost of the Head Office system.  -If that attribute is empty, attribute Cost will get information BasicCost Cost at the head Office Product Entity |
| **Category\_ID** | VARCHAR(11) |  |
| Category\_Name | NVARCHAR(50) |  |

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 --