STATIC PERSPECTIVE

Decomposition style and Layer style and Uses style  
  
**1. Primary presentation:**



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

|  |  |  |
| --- | --- | --- |
| Elements | | Properties |
| GUI | ProductUI | Display the product interface to the manager to control and perform some tasks such as add new product, update product, delete product. |
| CategoryUI | Display the category interface to manager to control and perform some task such as add new category, update category, delete category. Moreover, the manager can add new a product into a category or delete a product from category. |
| SynchUI | This interface which is modified synch of system by admin:   * Manual Synch: the cashier synchronizes data when needed. * Automatic Synch: the admin setting the synchronus time, and the system auto synchronized at a specified time. |
| UserAccount | This is a interface to manage user account. The admin can add new user, update or deactive user. |
| StatisticUI | This interface help the manager can make a statistics such as: a statistic about products which are bought per day/week/month/year; customers buy products on retail store frequently. |
| SalesUI | This interface help the cashier can pay a bill for customer. Select a payment method such as: pay by loyalty point mode or pay cash mode. |
| Business Logic | ProductBL | Includes functions to handle event that the user perform to do some operation related product. | |
| CategoryBL | Includes functions to handle event that the user perform to do some operation related category. | |
| StatisticBL | Includes functions to handle event that the user perform to do some operation related statistics. | |
| UserAccountBL | Includes functions to handle event that the user perform to do some operation related user account. | |
| SaleBL | Includes functions to handle event that the user perform to do some operation related sale. | |
| Common | This is the class containing some functions (or variable) are common to other classes in business logic layer. | |
| Data Transfer Object | ProductDTO | Includes variables only belong to Product (Examples: ProductID, ProductName, BasicCost, CategoryID), not include functions. | |
| CategoryDTO | Includes variables only belong to Category (Examples: CategoryID, CategoryName), not include functions. | |
| CustomerDTO | Includes variables only belong to Customer (Examples: CustomerID, CustomerName, CustomerAddress, CustomerPhone, Sumpoint), not include functions. | |
| BillDTO | Includes variables only belong to Bill (Examples: BillID, RetailStoreID, CustomerID, UserID, Date, TotalPoint, PlusPoint, MinusPoint), not include functions. | |
| BillDetailDTO | Includes variables only belong to BillDetail (Examples: BillID, ProductID, Quantity), not include functions. | |
| UserAccountDTO | Includes variables only belong to UserAccount (Examples: UserID, UserAddresss, UserName, UserPhone, Password, RetailStoreID), not include functions. | |
| Data Access | UserAccountDB | Includes function to access or store or update data to database related User Account | |
| ProductDB | Includes function to access or store or update data to database related Product | |
| CustomerDB | Includes function to access or store or update data to database related Customer | |
| CategoryDB | Includes function to access or store or update data to database related Category | |
| SaleDB | Includes function to access or store or update data to database related pay bill. | |

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 layer 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 separare following to N-tier, it looks like devide and conquer method. When we change some functions of this layer, then, it doesn’t affect much to other layers, so, maintenace 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, 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, Common.
* Data Access Layer: this layer are responsible to perform requirements of Business Logic Layer to database such as: update databse, read data from database and return to Business Logic Layer. This layer includes: UserDB, ProductDB, CustomerDB, CategoryDB, 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, UserDTO,.

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