Management store

Software Architecture Document

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 23/11/2019 | 1.0 | Architectural Goals and Constraints | Vo Khanh Vy |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

2. Architectural Goals and Constraints 4

3. Use-Case Model 4

4. Logical View 4

4.1 Component: abc 4

5. Deployment 4

6. Implementation View 4

Software Architecture Document

# Introduction

**Purpose**

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

**Scope**

This Software Architecture Document applies to the Management Store System which will be developed by Context Integration.

# Architectural Goals and Constraints

## Security

Central security features are handled by the institute officials. They will be given full access features both in the application and database levels. Creating user accounts for staff and warehouse staff are done by the admin staff. After the staff or warehouse staff are added to the system, they will be provided with a default password, which can be changed by the user. All the passwords are encrypted in order to ensure higher security. Responses and commits made by the staff and warehouse staff can be seen only by the admin.

## Persistence

All the data will be saved in the central server. This is a rational database that implements the Third normal form (MySQL). In order to maintain ACID (Atomicity, Confidentially, Integrity, Durability) some measures have been taken such as encrypting passwords, using transactions for all database commits,…

## Reliability / Availability

The system will be subjected to several tesing operations (Unit testing, integration testing, system tesing) before being deployed in order to make sure that the system is reliable. The MySQL database server can respond to many number of clients at a given moment without losing consistency and data integrity.

## Portability and reuse

Even though Management sales SW is as a complete and standalone product, it is possible to extend the produce to integrate in a system. In order to maintain reusability, all the wfunctionalities are very well structured and layered. Best practices of rup are followed throughout the project and the project strictly adheres to OOP standards

## Development tools

The project incoporates many development tools.

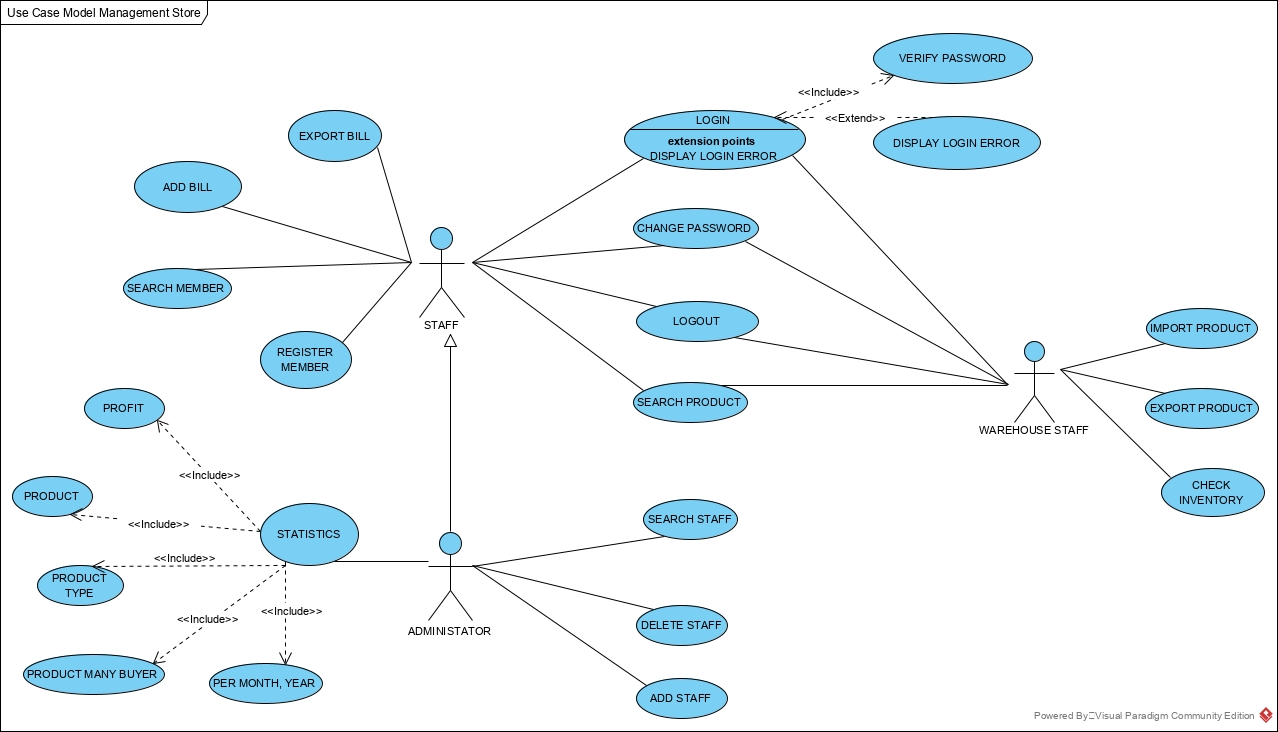
Programming:

Database:

Diagram: Draw.IO, PowerDesigner,…

Database connection: MYSQL connector

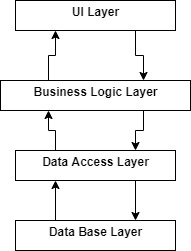
# Use-Case Model



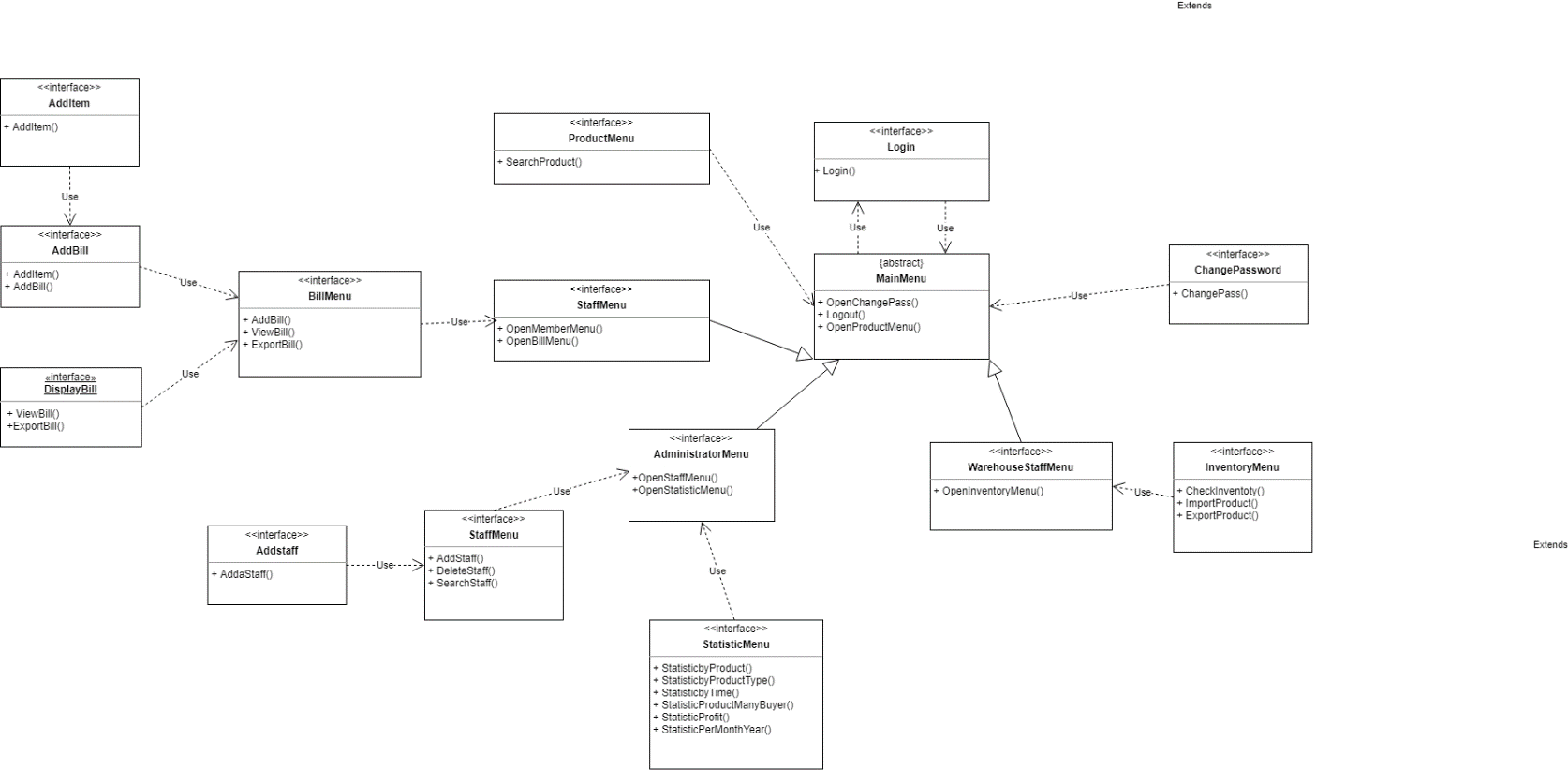
# Logical View

The logical view of the Convenience Store Sale Management Software follow the layered architecture with 4 main layer:

* **UI Layer**: Contains all the boundary classes that represent the software screens that the users use to communicate with the software.
* **Business Logic Layer**: Contains class that supports logic processing. It process user input from UI and transfer to the Data Access Layer. Process the data receive from the Data Access Layer and transfer it to UI.
* **Data Access Layer**: Contains class that supports access to the data base. It receive the statement from the Business Logic Layer and access to the data base toget the data. Transfer the data get from the data base to the Business Logic Layer.
* **Database Layer:** Contains the data base of the store.



## Component: UI Layer



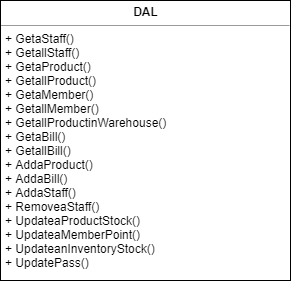
* Login class: Login screen class, supports login for all users.
* ChangePassword class: Change password screen, supports all users changing password.
* MainMenu class: Abstract classes for all user main menu classes.
* StaffMenu class, WarehouseStaffMenu, AdministratorMenu: Main menu screen for each type of staffs, warehouse staff, administrator. Displays menu to choose which action users want to do.
* BillMenu class: Menu displays for staffs and administrator all action can do on the bills.
* ProductMenu class: Displays screen support search and watch product information for all user.
* InventoryMenu class: Display screen supports warehouse staff choose to check inventory, to import or to export product.
* StaffMenu: Displays screen support administrator to add a staff, remove a staff or search for a staff.
* StatisticMenu: Displays screen support administrator to make statistic report in many type.

## Component: Business Logic Layer



* BLL class: Supports logical process in the software like process input from UI Layer for Data Access Layer or process data from Data Access Layer for UI.
* CheckLogin(): Check if username and password user input is right
* CheckMember(): Check if ID Member is exists
* CheckBill(): Check if ID Bill is exists
* CheckProduct(): Check if ID Product is exists
* CheckStaff(): Check if ID Staff is exists
* SearchMember(): Find member based on ID Member user input, if it exists, return information of this member, else show notice to user
* SearchBill(): Find bill based on ID Bill staff input, if it exists, return information of this bill, else show notice to staff
* SeachProduct(): Find product based on ID Product staff input, if it exists, return information of this product, else show notice to staff
* SearchStaff(): Find staff based on ID staff manager input, if it exists, return information of this staff, else show notice to manager
* StatisticProduct(): check amount of product in store/warehouse
* StatisticProductType(): check amount of type product in store/warehouse
* StatisticbyTime(): check the statistic of products sold over time
* StatisticProductManyBuyer(): statistic the products which bought many times
* StatisticProfit(): statistic revenue from sasles
* StatisticPerMotnYear(): statistic revenue by month of year
* AddaProducttoBill(): insert a product to bill
* AddBill(): insert a bill to system
* ExportaBill(): export a bill to the customer
* ImportProduct(): insert product to warehouse
* ExportProduct(): insert a product to store from warehouse
* AddaStaff(): add a new staff
* DeleteStaff(): remove a staff

## Component: Data Access Layer



* DAL class: Supports access to the data base to get data insert data, delete data, update data on the databse follow the require from the Business Logic Layer.
* GetaStaff(): Search information of a staff

+ Input: ID staff.  
+Output: all information of this staff.

* GetallSatff(): Search information all staff.

+Output: all information of all staff.

* GetaProduct(): Search information a product in store

+Input: ID product.  
+Output: all information of this product in store.

* GetallProduct(): Search information of all product

+Output: all in formation of all product in store.

* GetaMember(): Search information a member(customer)

+Input: ID member.

+Output: all information of this member.

* GetallMember(): Search information all member(customer)

+Output: all information of all Member.

* GetallProductinWarehouse(): Search information of all product which in Warehouse.

+Output: all information of all product in warehouse.

* GetaBill(): Search information a bill

+Input: ID bill.

+Output: information of this bill.

* GetallBill(): Search information of all bill

+Output: all information of all Bill.

* AddaProduct(): Add a new product into store

+Input: ID new product, name, value,…  
+Output: a new product inserted into database.

* AddaBill(): Add a new bill

+Input: ID new bill, ID staff, ID member,…  
+Output: a new bill inserted into database.

* AddaStaff(): Add a new staff

+Input: ID new staff, name, phone number,…

+Output: a new staff inserted into database.

* RemoveaStaff(): Delete a staff

+Input: ID staff.

+Output: this staff will be removed away from database.

* UpdateaProductStock(): Edit, update information of product

+Input: ID product.

+Output: this product will updated with new value.

* UpdateaMemberPoint(): Edit, update point of a member

+Input: ID member.

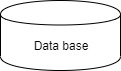
+Output: point of this member will be updated with new value.

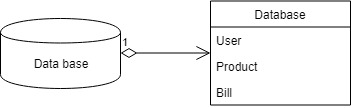
* UpdatePass(): all type of user can update their password

+For customer: update owner password.

+For staff: update owner password.  
+For manager: update owner password and other password of remain type of user.

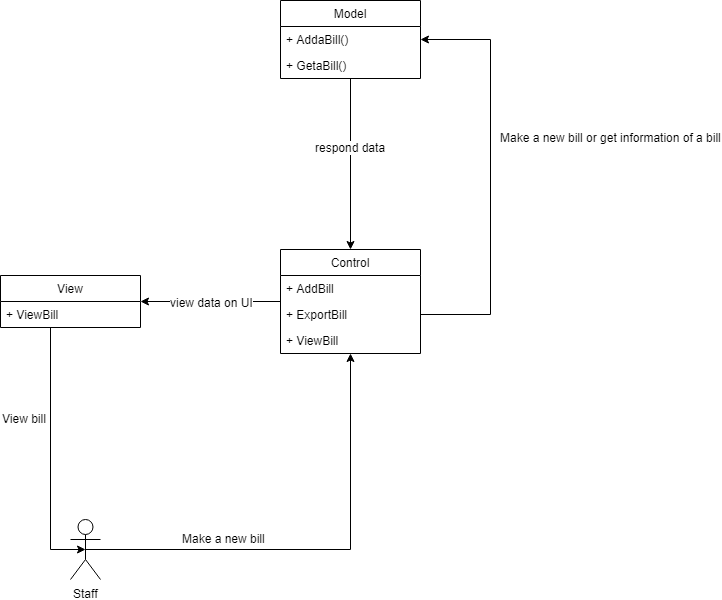
## Component: Database Layer



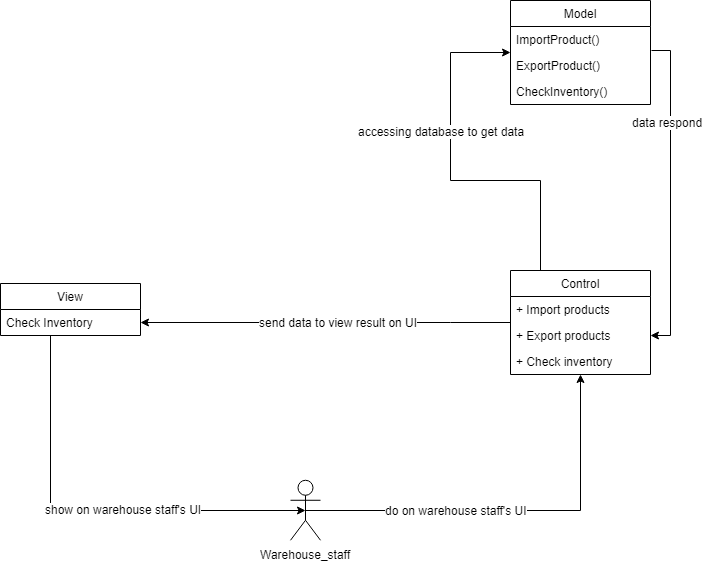


## Component: MVC model

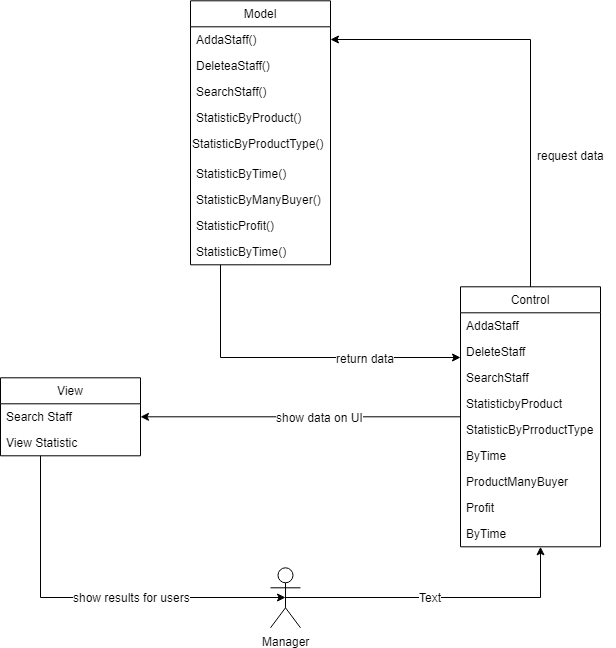
### For staff



### For warehouse staff



### For administrator (manager)



# Deployment

[Leave this section blank for PA3.]

# Implementation View

[Leave this section blank for PA3.]