**SOFTWARE DESIGN SPECIFICATION**



**SRM System**

**HIT Team**

Consulting

Sales

Staffing

Support

# Information of document

|  |  |
| --- | --- |
| **Title** | **Architecture 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** | Mr. Hung Anh |
| **Editor** |  |
| **Type of report** | Architecture 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 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

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. **Background and Scope**
      1. **Background**

Van Lang University (VLU) is facing with an urgent problem in the management of student records because the number of students is increasing. VLU was founded 15 years and has 15 Faculty / Department training distributed in two facilities. The total student population of about 12,000 students and university receives about 3,000 students yearly

Difficulties:

## When start the school year, VLU have to hire many external employees to input record in several days so the operation is done manually so it very difficult and consume time

## Faculties/ Departments cannot report the number of record received during the day, statistical reports must wait until the record receiving process end.

## Process using tools such as paper documents, Excel file, Word file to the manage records of all students will make it hard to search later

* + 1. **Scope**

Student come to VLU yearly to perform admission procedures, they must bringing matriculated paper and other records to VLU. Firstly student have to pay the tuition at accounting agent and then go through take photograph, finally bring all to the faculty to complete admission process

The system does not manage the paying process, taking pictures; only manage the record submission process in the faculty.

Customers Want:

* Report about the number of received record to date for human resources office or managing board can see updated information promptly.
* The input records will be encrypted to management software student information, do not need to input again.
* The finding information faster without losing time as searches in paper or Excel file
* Statistics in the form of selecting a date, selecting faculty or by the total number of passing students

Software product SRM is developed to solve problems in current processes, as well as help users manage the profile of student at the university more effectively.

* 1. **Stakeholders**

|  |  |  |
| --- | --- | --- |
| ID | Name | Description |
| E01 | Administrator | * To have full access to the system. * Manage all kind of user account * Import Student Record for School |
| E02 | The Faculty Monitor | * Manage user account works for their faculty * Analysis statistics and report about the number received student records of their faculty * Import Student Record for Faculty |
| E03 | The Received Student Record Officer | * Update student record of their faculty * Analysis statistics and report about the number received student records of their faculty * Import Student Record for Faculty |
| E04 | The Training Department Officer | * Import Student Record for School * Analysis statistics and report about the number received student records of school |
| E05 | The Human Resource Department Officer | * Analysis statistics and report about the number received student records of school |
| E06 | The Management Committee | * Analysis statistics and report about the number received student records of school |

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 POS Application | * It’s a WEB application |  |
| 5 | Database Server | * OS: Windows Server 2008 * Processor: 1 x Intel® Xeon® Processor E5606 * Memory: 1 x 2GB DDR3 1333 240- * Hard Disk: DELL 250GB SATA 7.2K 3.0Gbs 3.5" Enterprise * Software: Microsoft SQL Server 2008 Enterprise, .NET Framework 4.0 | * Run Main Database * Run Backup Database |
| 6 | WEB server | * Software: IIS 7.0 |  |
| 6 | User PC | * Operation System : Genuine Windows® 7 Home Basic, * Processor: AMD AM3 For Phenom™ II/Athlon™ II Family /Processors * Chipset: AMD SB710 * Graphics: ATI Radeon HD 3200, * Hard Drive SATA: 3.5" 320G * Software: Web browser | * Head office and Retail Store   + Run WEB SRM Application |
| 7 | Fiber router | * Vigor2950 | * Connect LAN with WAN |
| 8 | LAN | * Topology: Star * Use switch to connect elements in LAN | * Connect computers in a store |
| 11 | Fiber cable |  | * Connect fiber router in WAN by fiber port |

* + 1. **Relations and their properties**
    2. **Element behavior**
  1. **Architecture background**
     1. **Rationale design**

Deployment design satisfies quality attributes following:

Performance:

* + 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. **Layereds Style**



* + 1. **Uses Style**



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

|  |  |  |
| --- | --- | --- |
| Elements | | Properties |
| Controller | **StudentControllers** | The StudentController class contains action methods that render view pages. |
| **UserControllers** | The UserController class contains action methods that render view pages (AddUser, EditUser, ViewUser) |
| **StatisticsControllers** | The StatisticsController class contains action methods that render view pages (Statistics) |
| View | **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 the user to modify some information about the User. |
| **ViewStudent** | This GUI helps the user to view the Student. |
| **Statistics** | To make statistics about the total amount of Student following |
| Model | **StudentModels** | Storing and retrieving the Student information and return a message back to view pages. |
| **UserModels** | Storing and retrieving the User information and return a message back to view pages (AddUser, EditUser, ViewUser) |
| **StatisticsModels** | Storing and retrieving the some information related Statistics and return a message back to view pages (Statistics) |

* + - 1. **Layered Style**

|  |  |  |
| --- | --- | --- |
| Elements | | Properties |
| Presentation layer | **LogonUI** | User enter username and password at Logon screen. |
| **StudentMana**  **gementUI** | This GUI help user manage student record such as: import student record, edit student, view student list. |
| **StatisticsUI** | User can view student record following criteria. The student information are displayed on table. |
| Business Logic Layer | **PrintingBL** | This class have functions support printing receipt |
| **AuthorityBL** | This class contains functions that perform business logic for user management such as: add new user, edit user, view user. |
| **StudentBL** | Contains function that perform business logic for student record management such as: import student, view student, edit student |
| **Statistics\_BL** | Contains fucntion that help user to review list of student following criteria. |
| **LogonBL** | This class have functions that call functions from class User\_EF of Data Access Layer to check username and password of user. |
| **CommonBL** | Supports all class on Business Logic Layer by using common functions. |
| Data Access Layer | **UserEntity** | Consists of function to retrieve data from database or store, update data - which related User - to database. It is frequently called by class Authority**,** Logon. |
| **StudentEntity** | Consists of function to retrieve data from database or store, update data - which related Student - to database. It is frequently called by class Data Handle, Statistics. |
| **DepartmentEntity** | Consists of function to retrieve data from database or store, update data - which related Department - to database. It is frequently called by class Statistics, Data Handle. |
| **DbContext** | Context class is the primary class for interacting with data as objects that are instances of entity types that are defined in an Entity Data Model (EDM). |

* + 1. **Relations and their properties**

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

* 1. **Architecture background:**

The system has two GUI platforms: one is Window form that is separate into three layers such as: Presentation layer, Business Logic Layer, Data Access Layer; the other is Web form that uses MVC pattern such as: View, Model, Controller.

Let’s get to the bottom of three layers:

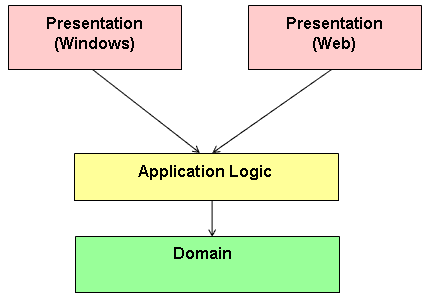
* Presentation layer: is responsible for communication with end user to collect data and show a result of data through components in user interface. We uses presentation layer for winfow form.
* Business Logic Layer contains components that are used to perform business logic and defines business entities that are used by the business component.
* Data Access Layer manages the physical storage and retrieval of data from database. We use the entity framework to Data Access Layer.
  + With Entity Framework, the developers issue queries using LINQ, then retrieve and manipulate data as strongly typed objects. The Entity Framework’s ORM implementation provides services like change tracking, identity resolution, lazy loading, and query translation so that developers can focus on their application-specific business logic rather than the data access fundamentals.
  + Its goal is to decrease the amount of code and maintenance required for data-oriented applications. Entity Framework applications provide the following benefits:
    - Applications can work in terms of a more application-centric conceptual model, including types with inheritance, complex members, and relationships.
    - Applications are freed from hard-coded dependencies on a particular data engine or storage schema.
    - Mappings between the conceptual model and the storage-specific schema can change without changing the application code.
    - Developers can work with a consistent application object model that can be mapped to various storage schemas, possibly implemented in different database management systems.
    - Multiple conceptual models can be mapped to a single storage schema.
    - Language-integrated query (LINQ) support provides compile-time syntax validation for queries against a conceptual model.

Besides using 3 layers, we also use MVC framework for Web form. They was using as GUI for client access by web browser. All data of system are taken by using web service to access Business Logic Layer. Let’s get to the bottom of MVC framework:

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

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

* *Views and controllers get connected automatically*. The MVC framework automatically establishes links between views and corresponding controllers. Therefore, the developers do not care about the associating views and controllers that linked to their views.
* *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. **Data Model**



|  |  |  |  |
| --- | --- | --- | --- |
| Entity | Attributes | Data Type | Description |
| Student | **Student\_ID** | VARCHAR(9) |  |
| Student \_Name | NVARCHAR(50) |  |
| Student \_Address | NVARCHAR(50) |  |
| Student \_Phone | INT |  |
| Student\_Gender | INT |  |
| RecordStatus\_Enrollment\_Notice | INT |  |
| RecordStatus\_HighSchool\_Transcript | INT |  |
| RecordStatus\_HighSchool\_Diploma(certificateBTTH,CNTNtemporary) | INT |  |
| RecordStatus\_Birth\_Certificate | INT |  |
| RecordStatus\_ThePaperCertifyingObject&PriorityAreas | INT |  |
| RecordStatus\_Matriculation\_Profile | INT |  |
| RecordStatus\_Permanent\_Residence | INT |  |
| Transfer\_Activity\_Profile | INT |  |
| RecordStatus\_4\_Stamped\_Envelope | INT |  |
| RecordStatus\_Photographed | INT |  |
| **Department\_ID** | VARCHAR(9) |  |
| **User\_ID** | VARCHAR(9) |  |
| User | **User\_ID** | VARCHAR(9) |  |
| User\_Name | NVARCHAR(50) |  |
| User\_Address | NVARCHAR(50) |  |
| User\_Phone | INT |  |
| Password | INT |  |
| User\_Gender | INT |  |
| **Department\_ID** | VARCHAR(9) |  |
| Department | **Department\_ID** | VARCHAR(9) |  |
| Department\_Name | NVARCHAR(50) |  |

1. **Dynamic Perspective**

**Component and Connector View**

* 1. **Primary presentation:**



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

|  |  |  |
| --- | --- | --- |
| **Elements** | | **Properties** |
| **Client tier** | **Web Browser** | A component that sends a request to web server and receive a reply is sent by web server. And then it displays UI to user. |
| **Student management** | This GUI help user manage student record such as: import student record, edit student, view student list. |
| **Statistics** | User can view student record following criteria. The student information are displayed on grid view. |
| **Database tier** | **Primary Database** | It’s main database server which uses frequently when the system work properly. It’s responsible for store data such as sales data, user data, customer data, store data, product data, and category data. In the certain time, it will synch with temporary database as a backup data. |
| **Backup Database** | It’s a temporary database which uses rarely. It only uses when the system doesn’t work properly, crash or not available. In the certain time, it will synch all data with primary. |
| **Server Tier** | **StudentControllers** | The StudentController class contains action methods that render view pages. |
| **UserControllers** | The UserController class contains action methods that render view pages (related AddUser, EditUser, ViewUser) |
| **StatisticsControllers** | The StatisticsController class contains action methods that render view pages (related Statistics) |
| **Web Service** | Web service is a method of communication between two electronic devices via the Web (Internet). It is designed to support [interoperable](http://en.wikipedia.org/wiki/Interoperability) machine-to-machine interaction over a [network](http://en.wikipedia.org/wiki/Computer_network)". It uses [SOAP](http://en.wikipedia.org/wiki/SOAP) protocol, typically conveyed by using [HTTP](http://en.wikipedia.org/wiki/HTTP) with an [XML](http://en.wikipedia.org/wiki/XML) [serialization](http://en.wikipedia.org/wiki/Serialization) in conjunction with other Web-related standards. |
| **IIS 7** | IIS 7 is a major enhancement to the Windows web platform and plays a central role in unifying Microsoft web platform technologies - ASP.NET, Windows Communication Foundation web services, and Windows SharePoint Services.  It handle requests and replies that are send between client tier and server tier. |
| **ViewUser** | This GUI helps the user to view the User. |
| **AddUser** | This GUI helps the user to add a new User. |
| **ViewStudent** | This GUI helps the user to view the Student. |
| **UserModels** | Storing and retrieving the User information and return a message back to view pages (AddUser, EditUser, ViewUser) |
| **StudentModels** | Storing and retrieving the Student information and return a message back to view pages. |
| **AuthorityBL** | This class contains functions that perform business logic for user management such as: add new user, edit user, view user. |
| **StudentBL** | Contains function that perform business logic for student record management such as: import student, view student, edit student |
| **StatisticsBL** | Contains function that help user to review list of student following criteria. |
| **UserEntity** | Consists of function to retrieve data from database or store, update data - which related User - to database. It is frequently called by class Authority**,** Logon. |
| **StudentEntity** | Consists of function to retrieve data from database or store, update data - which related Student - to database. It is frequently called by class Data Handle, Statistics. |
| **DepartmentEntity** | Consists of function to retrieve data from database or store, update data - which related Department - to database. It is frequently called by class Statistics, Data Handle. |
| **TempDB.xml** | When client send a request to get data from DB to display UI. The system will store this data in TempDB as a temporary data. Afterwards, if client need this data, and sent request to Data Access, it will read TempDB file and return data to client to display UI, needless to access database. |

* + 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 and return** | Responsible for conveying the service request from the requester to the provider and for returning any results. Use by interface to request data from Filter/Object |
| **Synchronous Replication** | The technique for replicating data by two or more databases (or file systems) where the system being replicated does waits for the data to have been recorded on the duplicate system before proceeding. |
| **Soap/Http** | Stand for Simple Object Access Protocol. It is the kind of protocol for communication in Web Service. It chooses XML for a standard message format. Usually, it relies on other [Application Layer](http://en.wikipedia.org/wiki/Application_Layer) protocols, mostly HTTP. |
| **File I/O** | Refer the communication in access data in a file to perform operation such as: read file and write file. |
| **ODBC** | Stands for Open Database Connectivity. It is the standard method which allows any application to connect data. ODBC uses a middle layer called the database driver to handles the connection in between the application and the relational database management system. |

.

1. **Solution background**
   1. **Architectural Approaches**
2. **Mapping between perspectives** 
   1. **Mapping between a module view and a component-and-connector view**
   2. **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 |

## -- The End --