**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

**Computer Product Suggestion**

|  |  |
| --- | --- |
| **Group 13** | |
| **Group member** | Pham Hong Sang – Team Leader – SE60601  Huynh Thanh Viet – Team Member - SE60666  Tran Tan Len – Team Member - SE60623  Ha Chi Danh – Team Member – 60431 (Dropped out) |
| **Supervisor** | Mr. Kieu Trong Khanh |
| **Ext. Supervisor** | N/A |
| **Capstone Project code** | CPS |

-Ho Chi Minh City, 05/2014-

*This page is intentionally left blank*

***ACKNOWLEDGEMENTS***

We wish to thank various people for their contribution to this project: Our teachers for their advice and participation in the final review, our friends for their valuable technical support.

Special thanks should be given to Mr. Kieu Trong Khanh, our research supervisor for his professional guidance and the useful, constructive recommendations throughout the course of this project.

# Table of Contents

[Table of Contents 4](#_Toc395826427)

[A. Software Project Management Plan 6](#_Toc395826428)

[1. Problem Definition 6](#_Toc395826429)

[1.1 Name of this Capstone Project 6](#_Toc395826430)

[1.2 Problem Abstract 6](#_Toc395826431)

[1.3 Current Comparison websites in Vietnam 6](#_Toc395826432)

[1.4 The Proposed System 6](#_Toc395826433)

[2. Solution đang phân vân 2 cái máy tính 7](#_Toc395826434)

[3. Project organization 7](#_Toc395826435)

[3.1 Software Process Model 7](#_Toc395826436)

[3.2 Tools and Techniques 7](#_Toc395826437)

[4. System Architectural Design 9](#_Toc395826438)

[4.1 Web Architecture 9](#_Toc395826439)

[5. Component Diagram 9](#_Toc395826440)

[6. Conceptual Diagram 10](#_Toc395826441)

[7. Entity Relationship Diagram 10](#_Toc395826442)

[8. Class Diagram 17](#_Toc395826443)

[8.1 Class Diagram Explanation 17](#_Toc395826444)

[9. Database Relationship Diagram 24](#_Toc395826445)

[9.1 Diagram 24](#_Toc395826446)

[9.2 Data Dictionary 25](#_Toc395826447)

[10. Sequence Diagram 28](#_Toc395826448)

[10.1 Force Parse Data 29](#_Toc395826449)

[10.2 Import Product 29](#_Toc395826450)

[10.3 Process Error Products 29](#_Toc395826451)

[10.4 Process Duplicated Products 30](#_Toc395826452)

[10.5 Save Correct Products 30](#_Toc395826453)

[10.6 Log File 31](#_Toc395826454)

[10.7 Recommend 32](#_Toc395826455)

[10.8 View History 32](#_Toc395826456)

[10.9 Search Product 32](#_Toc395826457)

[10.10 Configure System 33](#_Toc395826458)

[10.11 Confirm New Product 33](#_Toc395826459)

[10.12 Training Machine 34](#_Toc395826460)

[11. Algorithms 35](#_Toc395826461)

[11.1 Calculating Product Point 35](#_Toc395826462)

[11.2 String Comparison 35](#_Toc395826463)

[11.2.1.1 Define 35](#_Toc395826464)

[12. Demo 37](#_Toc395826465)

[12.1 Scenario 37](#_Toc395826466)

[13. Advantages and disadvantages 37](#_Toc395826467)

[13.1 Advantages 37](#_Toc395826468)

[13.2 Disadvantages 37](#_Toc395826469)

# Software Project Management Plan

## Problem Definition

### Name of this Capstone Project

Computer Product Suggestion (CPS).

### Problem Abstract

* Nowadays Online shopping is become the most popular trends in the world.E-commerce websites are become more and more popular; however, they just only show the details of products and don’t have any effective search and compare function that can recommend for customer about their choices.
* But, how can we know a computer is better than the other ones? Or which one is fixed their budget? Our system will do that thing, it helps users find computer online, compares them and system will suggest products what is suitable for users, etc…

### Current Comparison websites in Vietnam

Below are some comparison sites:

* Normal E-commerce websites (thegioididong.com, vienthonga.com, dienmay.com, etc…): They have some functions that let people search and see details of each product.
* But all that products are had in their website and we can’t compare them with another website. They show all text details, it is too difficult for users to choose what they want and it spends too much time.
* Especial compare websites (compare.vn, websosanh.vn, sosanh.vn, etc…): They provide functions that let users add 2 or more in order to make them see details of products easily. They collect data from another website, so that make users see more details of product then a normal e-commerce websites. But they still have no any especial search and compare functions.

### The Proposed System

The system is intended for users to make decisions about set of computer products that they want to buy. The system must to manage products, users, etc… In detail, the system will enable following function:

* Admins can manage the system, manage accounts, and configure system.
* System can evaluate the inputted product to give suggestion or proposal, beside that it will parse the web to get the useful information.
* Staff will define or configure the weight of criteria and collect data from web to mine.
* Users can request to search and get the suggestion with set of selected products and recommend and rating for each product.
* Trainings module will help system recognize products are already exist in database or not. If not, system will be trained about products.

## Solution đang phân vân 2 cái máy tính

## Project organization

### Software Process Model

Project is developed under agile model.



Figure 1: Agile Development Model

For more information: <http://www.indicthreads.com/1439/quick-introduction-to-agile-software-development/>

(Owner: IndicThreads.com. Online Software Developer Magazine and Conferences)

### Tools and Techniques

- Front-end technologies: HTML5, CSS3, JavaScript, jQuery, AJAX.

- Back-end:

* Website: ASP.NET MVC4 + Entity Framework 5.
* Scheduler: Quartz.
* Parse data from Excel file: OLEDB.NET.

- Web Server: Microsoft IIS.

- Database Management System: MS SQL Server 2008 Enterprise R2.

## System Architectural Design

### Web Architecture

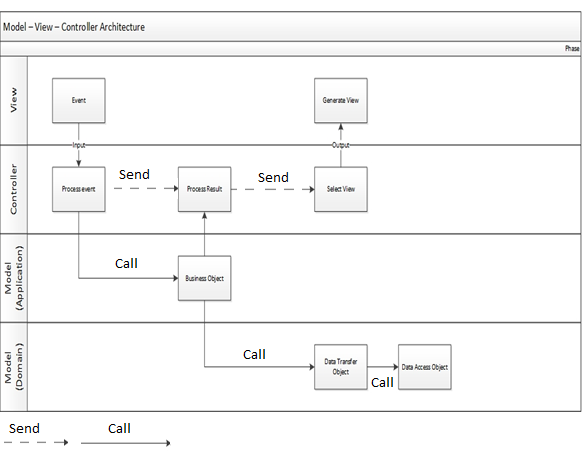


Figure 2: MVC Architecture

* **Model** is the part of the application that handles the logic for the application data. Often model objects retrieve data (and store data) from a database.
* **View** is the parts of the application that handles the display of the data. Most often the views are created from the model data.
* **Controller** is the part of the application that handles user interaction. Typically controllers read data from a view, control user input, and send input data to the model.

## Component Diagram

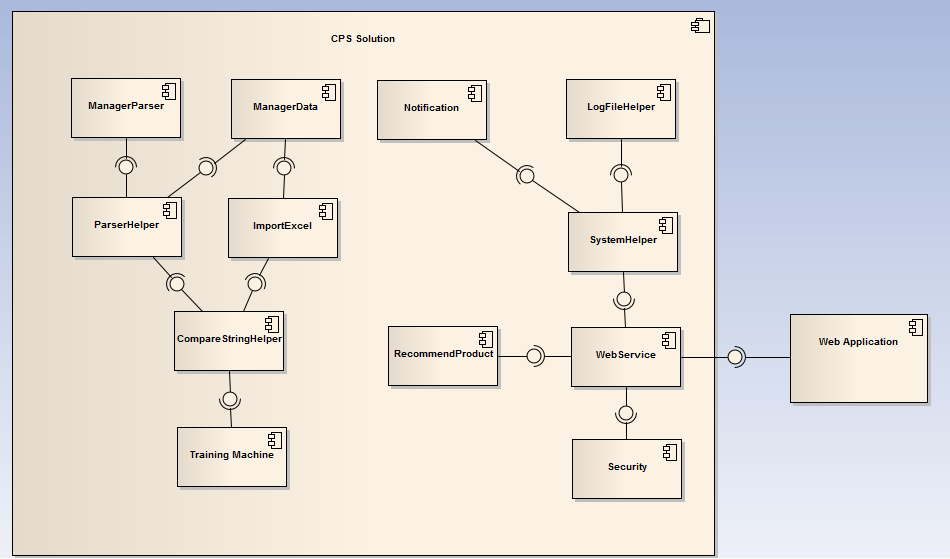
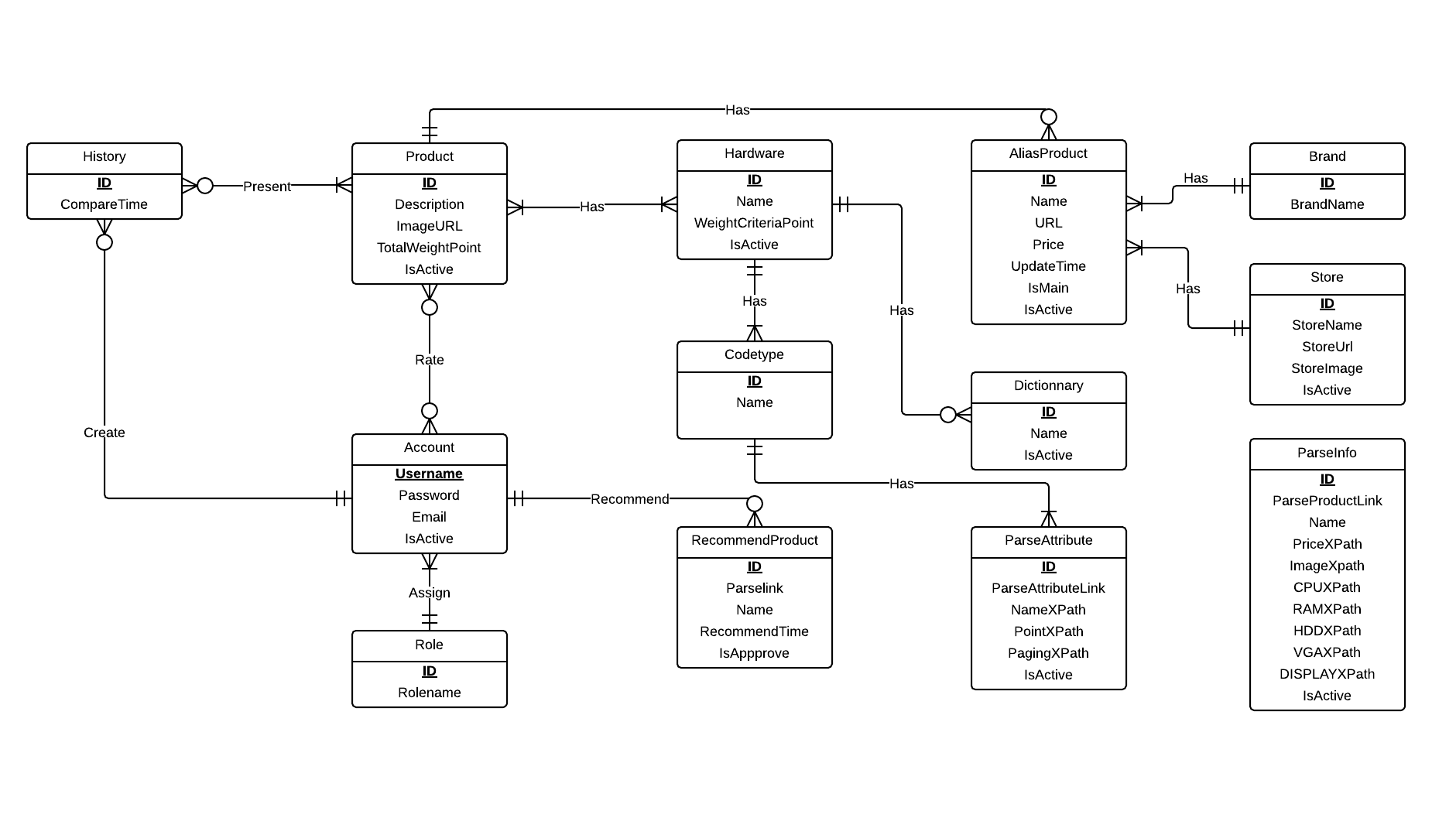


Figure 3: Component Diagram

## Conceptual Diagram



## Entity Relationship Diagram

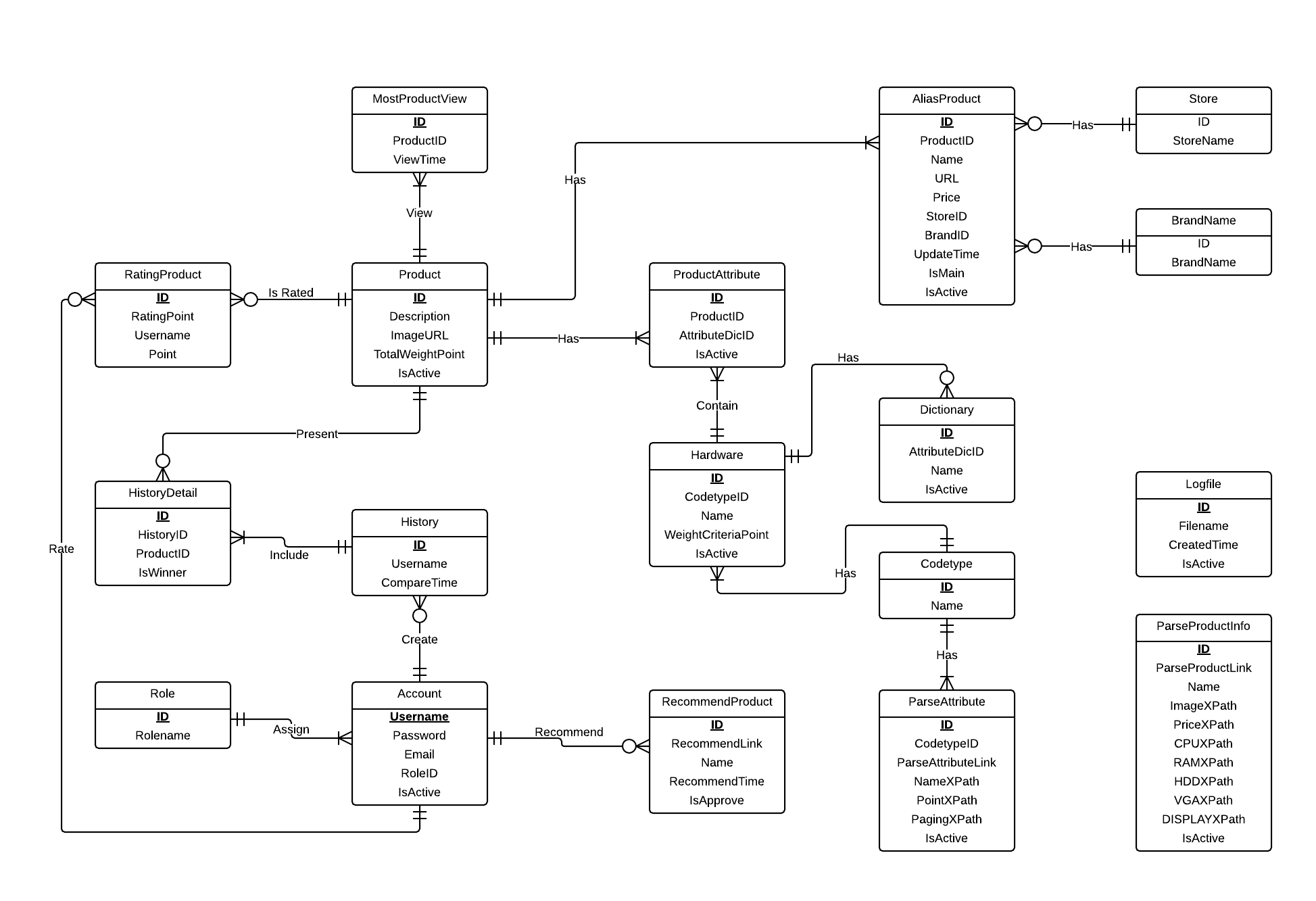


Figure 4: Entity Relationship Diagram

#### Data Dictionary

|  |  |
| --- | --- |
| **Entity Data dictionary: describe content of all entities** | |
| **Entity Name** | **Description** |
| Codetype | Describe all codetype in the system. |
| ParseInfo | Describe all data needed to parse 1 specified product |
| Hardware | Describe all hardwares of a product. |
| Product | Describe all products in the system. |
| ProductAttribute | Describe relationship between Attribute and Product. |
| AliasProdcut | Describe all alias name of product in the system. |
| Dictionary | Describe all possible hardwares names in the system. |
| Account | Describe all accounts in the system. Account includes: admin, staff, member, guest… |
| RecommendProduct | Describe the product that user recommend for system. |
| History | Describe all compare histories of user in the system. |
| HistoryDetail | Describe details of any history in the system. |
| Role | Describe all roles in the system. One user has only one role. |
| LogFile | Describe all log files of the system. |
| ParserAttribute | Describe all data needed to parse 1 websites to get attribute |
| RatingProduct | Describe relationship between Attribute and Product. |
| Brand | Describe all brand that product is included. |
| Store | Describe all store that product is included. |
| MostViewProduct | Describe how many time that 1 products are viewed. |

Table 1: Entity Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity name** | **Attributes** | **Description** | **Domain** | **Null** |
| Market | Id {PK} | Unique identifier of a market, auto increment. | Integer | No |
| Name | Market name. | Unicode string, length: 50 | No |
| Address | Market address. | Unicode string, length: 100 | Yes |
| Latitude | Market latitude. | Float | Yes |
| Longitude | Market longitude. | Float | Yes |
| IsActive | Market status. | Bit | No |
| Parse Info | Id {PK} | Unique identifier of a parser, auto increment. | Integer | No |
| MarketId | Market identifier. | Integer | No |
| ParseLink | Link to parse. | Unicode string, length: 300 | No |
| ProductNameXpath | Xpath to get product name. | Unicode string, length: 100 | No |
| PriceXpath | Xpath to get price. | Unicode string, length: 100 | No |
| PagingXpath | Xpath to get paging part. | Unicode string, length: 100 | No |
| IsActive | Parser status. | Bit | No |
| Product | Id {PK} | Unique identifier of a parser, auto increment. | Integer | No |
| Name | Product name. | Unicode string, length: 100 | No |
| IsActive | Product status. | Bit | No |
| Product Attribute | Id {PK} | Unique identifier, auto increment. | Integer | No |
| ProductId | Product identifier. | Integer | No |
| MinPrice | Product min price. | Integer | No |
| MaxPrice | Product max price. | Integer | No |
| LastUpdatedTime | The time this record is inserted. | Datetime | No |
| Log File | Id {PK} | Unique identifier of a log file, auto increment. | Integer | No |
| FileName | The file name. | String, length: 100 | No |
| CreatedTime | The time that file is created. | Datetime | No |
| IsActive | The file status. | Bit | No |
| User | Username {PK} | The name chosen by user. | String, length: 50 | No |
| Password | Password of an account. | String, length: 20 | No |
| IsActive | Account status. | Bit | No |
| Role | Id {PK} | Unique identifier a role, auto increment. | Integer | No |
| Name | Role name. | Unicode string, length: 50 | No |
| Profile | Username {PK} | Owner of this profile. | String, length: 50 | No |
| FirstStartAddress | Start address of the first route. | Unicode string, length: 100 | Yes |
| FirstEndAddress | End address of the first route. | Unicode string, length: 100 | Yes |
| FirstRoute | First favourite route. | Unicode string, length: 200 | Yes |
| FirstMarkets | Nearby market on the first route. | Unicode string, length: 100 | Yes |
| FirstStartDistance | All distance from first start point to all nearby markets. | String, length: 150 | Yes |
| FirstEndDistance | All distance from first end point to all nearby markets. | String, length: 150 | Yes |
| FirstRouteName | Name of the first route. | Unicode string, length: 50 | Yes |
| SecondStartAddress | Start address of the second route. | Unicode string, length: 100 | Yes |
| SecondEndAddress | End address of the second route. | Unicode string, length: 100 | Yes |
| SecondRoute | Second favourite route. | Unicode string, length: 200 | Yes |
| SecondMarkets | Nearby market on the second route. | Unicode string, length: 100 | Yes |
| SecondStartDistance | All distance from second start point to all nearby markets. | String, length: 150 | Yes |
| SecondEndDistance | All distance from second end point to all nearby markets. | String, length: 150 | Yes |
| SecondRouteName | Name of the second route. | Unicode string, length: 50 | Yes |
| ThirdStartAddress | Start address of the third route. | Unicode string, length: 100 | Yes |
| ThirdEndAddress | End address of the third route. | Unicode string, length: 100 | Yes |
| ThirdRoute | Third favourite route. | Unicode string, length: 200 | Yes |
| ThirdMarkets | Nearby market on the third route. | Unicode string, length: 100 | Yes |
| ThirdStartDistance | All distance from third start point to all nearby markets. | String, length: 150 | Yes |
| ThirdEndDistance | All distance from third end point to all nearby markets. | String, length: 150 | Yes |
| ThirdRouteName | Name of the third route. | Unicode string, length: 50 | Yes |
| Market Distance | FromMarket {PK} | Unique identifier of a market. | Integer | No |
| ToMarket {PK} | Unique identifier of a market. | Integer | No |
| Distance | Distance between two markets. | Float | No |
| History | Id {PK } | Unique identifier of a history. | Integer | No |
| Username | Owner of this history. | String, length: 50 | No |
| BuyTime | The time this history is created. | Datetime | No |
| History Detail | Id {PK} | Uniquely identifier, auto increment. | Integer | No |
| HistoryId | History identifier. | Integer | No |
| ProductId | Product identifier. | Integer | No |
| MinPrice | Product min price at the time it was bought. | Integer | No |
| MaxPrice | Product max price at the time it was bought. | Integer | No |
| Dictionary | Id {PK} | Unique identifier of a word, auto increment. | Integer | No |
| Name | Possible name of a product. | Unicode string, length: 100 | No |
| ProductId | Product identifier. | Integer | No |
| Sell Product | Id {PK } | Unique identifier, auto increment. | Integer | No |
| MarketId | Market identifier. | Integer | No |
| ProductId | Product identifier. | Integer | No |
| SellPrice | The price at which the product is sold. | Integer | No |
| LastUpdatedTime | The time data was last updated. | Datetime | No |
| User Price | Id {PK} | Unique identifier, auto increment. | Integer | No |
| Username | User who proposed the price. | String, length: 50 | No |
| MarketId | Market identifier. | Integer | No |
| ProductId | Product identifier. | Integer | No |
| UpdatedPrice | The price user proposed. | Integer | No |
| LastUpdatedTime | The time user proposed price. | Datetime | No |

Table 2: Attribute Data Dictionary

## Class Diagram

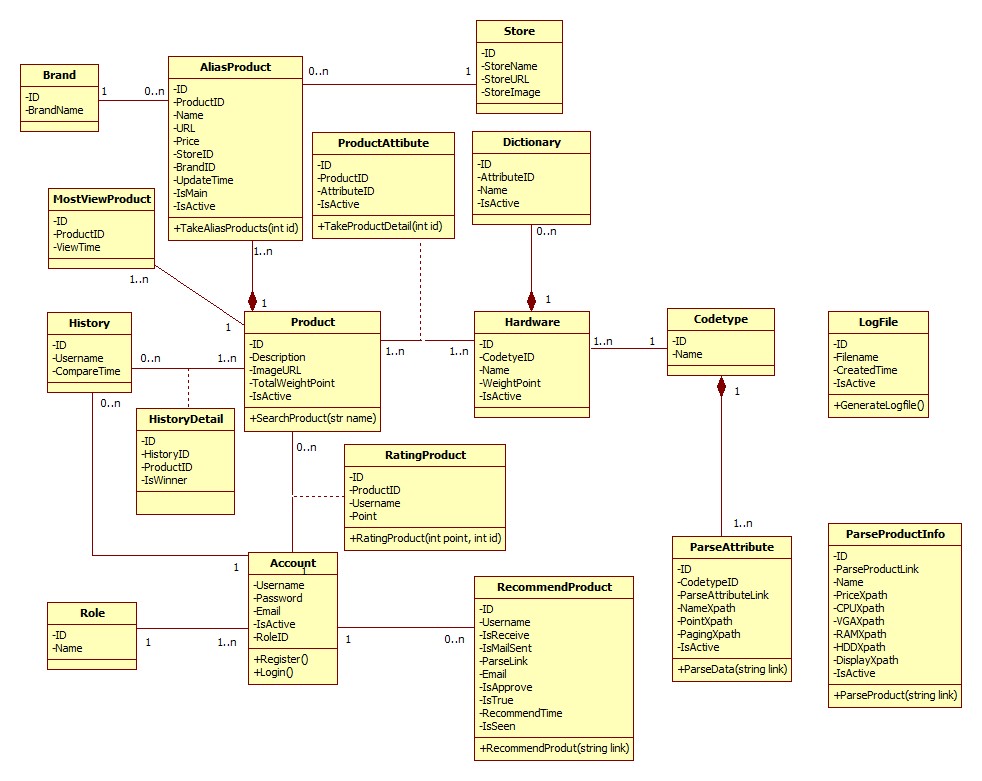


Figure 5: Class Diagram

### Class Diagram Explanation

#### Codetype

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each code type |
| Name | String | Public | Name of code type |
| IsActive | Boolean | Public | Status of code type |

#### ParseAttribute

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each item |
| CodetypeID | String | Public | Id of codetype |
| ParseAttributeLink | String | Public | Link used to parse |
| NameXpath | String | Public | Xpath to get the name of attribute |
| PointXpath | String | Public | Xpath to get the point of attribute |
| PagingXpath | String | Public | Xpath to get the paging |
| IsActive | Boolean | Public | Status of attribute |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| ParseData | Void | Public | Run parser to get data |

#### Hardware

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each item |
| CodetypeId | Integer | Public | Id of codetype |
| Name | String | Public | Name of attribute |
| WeightCriteriaPoint | Integer | Public | Point of each hardware |
| IsActive | Boolean | Public | Status of Hardware |

#### Dictionary

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each product |
| AttributeID | Integer | Public | Id of attribute |
| Name | String | Public | Name of attribute |
| IsActive | Boolean | Public | Status of Dictionnary |

#### Product

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each item |
| Description | String | Public | Description of this product |
| ImageURL | String | Public | ImageURL of this product |
| TotalWeightPoint | Double | Public | Point of this product |
| IsActive | Boolean | Public | Status of product |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Search Product | Product | Public | Search product in system |

#### MostViewProduct

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each item |
| ProductID | String | Public | ID of each Product |
| ViewTime | Int | Public | View time of each product. |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| LoadMostProduct | Product | Public | Load most view product |

#### Product Attribute

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each item |
| ProductID | Int | Public | Id of product |
| AttributeID | Int | Public | Id of attribute |
| IsActive | Boolean | Public | Status of each attribute |

#### AliasProduct

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ID | Int | Public | Unique identifier of each item |
| ProductID | Int | Public | Id of product |
| Name | String | Public | Product name |
| URL | String | Public | URL of product |
| Price | Double | Public | Price of each produt |
| StoreID | Int | Public | Id of store |
| BrandID | Int | Public | Id of brand |
| UpdateTime | Datetime | Public | Time update that product |
| IsMain | Boolean | Public | Describe which main product is. |
| IsActive | Boolean | Public | Status of this productalias |

#### History

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each history |
| Username | String | Public | Owner of this history |
| CompareTime | DateTime | Public | The time when owner compare product |

#### History Detail

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each item |
| HistoryId | Integer | Public | Id of history |
| ProductId | Integer | Public | Id of product |
| IsWinner | Boolean | Public | Winner Product |

#### Role

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each role |
| Name | String | Public | Role name |

#### Log File

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each file |
| FileName | String | Public | File name |
| CreatedTime | DateTime | Public | The time when this file is created |
| IsActive | Boolean | Public | Status of this file |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| GenerateLogFile | Boolean | Public | Generate log file for each time system runs the parser |

#### Rating Product

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique id of each item |
| Username | String | Public | The name of the user |
| ProductId | Integer | Public | The identifier of that product |
| Point | Integer | Public | Rating point of each product |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| RatingProduct | Void | Public | rating price for the system |

#### Recommend Product

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique id of each item |
| Username | String | Public | The name of the user |
| IsReceive | Boolean | Public | Receive mail when update product/ |
| IsMailSent | Boolean | Public | Send email or not yet. |
| Parselink | String | Public | Link that user recommend |
| Email | String | Public | Email of user |
| IsApprove | Boolean | Public | Status of recommend |
| IsTrue | Boolean | Public | Status of auto parse recommend. |
| RecommendTime | Datetime | Public | Time that user recommend |
| IsSeen | Boolean | Public | Status of notification for user |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| RecommendProduct | Void | Public | Recommend product for system |

#### Parse Product Info

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique id of each item |
| ParseLink | String | Public | Link to get parser |
| Name | String | Public | Xpath to get the Name |
| PriceXpath | String | Public | Xpath to get the Price |
| ImageXpath | String | Public | Image to get the Name |
| CPUXpath | String | Public | Xpath to get the CPU |
| HDDXpath | String | Public | Xpath to get the HDD |
| VGAXpath | String | Public | Xpath to get the VGA |
| RAMXpath | String | Public | Xpath to get the RAM |
| DisplayXpath | String | Public | Xpath to get the Display |
| IsActive | String | Public | Status of parser |

#### Account

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Username | String | Public | Unique username of each user |
| Password | String | Public | Password of user |
| Email | String | Public | Email of User |
| RoleId | Int | Public | Account roles ID |
| IsActive | String | Public | Status of account |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Register | Boolean | Public | Register a new account in system |
| Login | Boolean | Public | Check if is existed account in system |

#### Store

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ID | Int | Public | Unique of each store |
| StoreName | String | Public | Name of store |
| StoreUrl | String | Public | Url of each store |
| LogoImage | String | Public | Image URL of each store |
| IsActive | String | Public | Status of store |

#### Brand

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ID | Int | Public | Unique of each Brand |
| BrandName | String | Public | Name of each Brand |

## Database Relationship Diagram

### Diagram

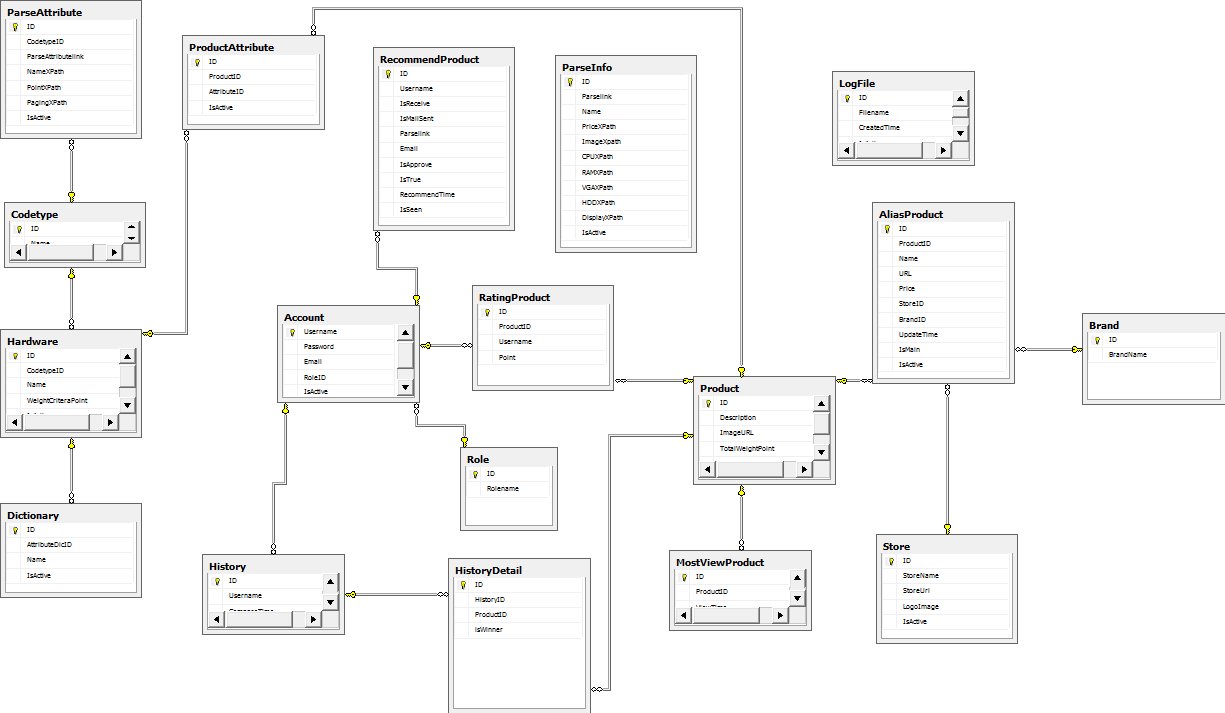


Figure 6: Database Relationship Diagram

### Data Dictionary

|  |  |
| --- | --- |
| **Entity Data dictionary: describe content of all entities** | |
| Entity Name | Description |
| Market | Describe all markets in the system. Market may exist in real life or just virtual. |
| Parse Info | Describe all needed data to parse a website. |
| Product | Describe all products in the system. |
| Product Attribute | Describe attributes of each product. |
| Log File | Describe all log files of the system. Log file is generated after parsers finish running. |
| User | Describe all accounts in the system. |
| Role | Describe all roles in the system. |
| Profile | Describe all user profiles in the system. Each user has only one profile. |
| Market Distance | Describe distance between two markets. |
| History | Describe all users’ buying history. |
| History Detail | Describe all details of each history. |
| Dictionary | Describe all possible product names in the system. |
| Sell Product | Describe product price at each market. |
| User Price | Describe all users’ proposed price. |

Table 3: Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity name** | **Attributes** | **Description** | **Domain** | **Null** |
| Market | Id {PK} | Unique identifier of a market, auto increment. | int | No |
| Name | Market name. | nvarchar(50) | No |
| Address | Market address. | nvarchar(100) | Yes |
| Latitude | Market latitude. | float | Yes |
| Longitude | Market longitude. | float | Yes |
| IsActive | Market status. | bit | No |
| Parse Info | Id {PK} | Unique identifier of a parser, auto increment. | int | No |
| MarketId {FK} | Foreign key, market identifier. | int | No |
| ParseLink | Link to parse. | nvarchar(300) | No |
| ProductNameXpath | Xpath to get product name. | nvarchar(100) | No |
| PriceXpath | Xpath to get price. | nvarchar(100) | No |
| PagingXpath | Xpath to get paging part. | nvarchar(100) | No |
| IsActive | Parser status. | bit | No |
| Product | Id {PK} | Unique identifier of a parser, auto increment. | int | No |
| Name | Product name. | nvarchar(100) | No |
| IsActive | Product status. | bit | No |
| Product Attribute | Id {PK} | Unique identifier, auto increment. | int | No |
| ProductId {FK} | Foreign key, product identifier. | int | No |
| MinPrice | Product min price. | int | No |
| MaxPrice | Product max price. | int | No |
| LastUpdatedTime | The time this record is inserted. | datetime | No |
| Log File | Id {PK} | Unique identifier of a log file, auto increment. | int | No |
| FileName | The file name. | varchar(100) | No |
| CreatedTime | The time that file is created. | datetime | No |
| IsActive | The file status. | bit | No |
| User | Username {PK} | The name chosen by user. | varchar(50) | No |
| Password | Password of an account. | varchar(20) | No |
| RoleId {FK} | Foreign key, role identifier. | int | No |
| IsActive | Account status. | bit | No |
| Role | Id {PK} | Unique identifier a role, auto increment. | int | No |
| Name | Role name. | nvarchar(50) | No |
| Profile | Username {PK, FK} | Owner of this profile. | varchar(50) | No |
| FirstStartAddress | Start address of the first route. | nvarchar(100) | Yes |
| FirstEndAddress | End address of the first route. | nvarchar(100) | Yes |
| FirstRoute | First favourite route. | nvarchar(200) | Yes |
| FirstMarkets | Nearby market on the first route. | nvarchar(100) | Yes |
| FirstStartDistance | All distance from first start point to all nearby markets. | varchar(150) | Yes |
| FirstEndDistance | All distance from first end point to all nearby markets. | varchar(150) | Yes |
| FirstRouteName | Name of the first route. | nvarchar(50) | Yes |
| SecondStartAddress | Start address of the second route. | nvarchar(100) | Yes |
| SecondEndAddress | End address of the second route. | nvarchar(100) | Yes |
| SecondRoute | Second favourite route. | nvarchar(200) | Yes |
| SecondMarkets | Nearby market on the second route. | nvarchar(100) | Yes |
| SecondStartDistance | All distance from second start point to all nearby markets. | varchar(150) | Yes |
| SecondEndDistance | All distance from second end point to all nearby markets. | varchar(150) | Yes |
| SecondRouteName | Name of the second route. | nvarchar(50) | Yes |
| ThirdStartAddress | Start address of the third route. | nvarchar(100) | Yes |
| ThirdEndAddress | End address of the third route. | nvarchar(100) | Yes |
| ThirdRoute | Third favourite route. | nvarchar(200) | Yes |
| ThirdMarkets | Nearby market on the third route. | nvarchar(100) | Yes |
| ThirdStartDistance | All distance from third start point to all nearby markets. | varchar(150) | Yes |
| ThirdEndDistance | All distance from third end point to all nearby markets. | varchar(150) | Yes |
| ThirdRouteName | Name of the third route. | nvarchar(50) | Yes |
| Market Distance | FromMarket {PK, FK} | Unique identifier of a market. | int | No |
| ToMarket {PK, FK} | Unique identifier of a market. | int | No |
| Distance | Distance between two markets. | float | No |
| History | Id {PK } | Unique identifier of a history. | int | No |
| Username {FK} | Foreign key, owner of this history. | varchar(50) | No |
| BuyTime | The time this history is created. | datetime | No |
| History Detail | Id {PK} | Uniquely identifier, auto increment. | int | No |
| HistoryId {FK} | Foreign key, history identifier. | int | No |
| ProductId {FK} | Foreign key, product identifier. | int | No |
| MinPrice | Product min price at the time it was bought. | int | No |
| MaxPrice | Product max price at the time it was bought. | int | No |
| Dictionary | Id {PK} | Unique identifier of a word, auto increment. | int | No |
| Name | Possible name of a product. | nvarchar(100) | No |
| ProductId {FK} | Foreign key, product identifier. | int | No |
| Sell Product | Id {PK } | Unique identifier, auto increment. | int | No |
| MarketId {FK} | Foreign key, market identifier. | int | No |
| ProductId {FK} | Foreign key, product identifier. | int | No |
| SellPrice | The price at which the product is sold. | int | No |
| LastUpdatedTime | The time data was last updated. | datetime | No |
| User Price | Id {PK} | Unique identifier, auto increment. | int | No |
| Username {FK} | Foreign key, user who proposed the price. | varchar(50) | No |
| MarketId {FK} | Foreign key, market identifier. | int | No |
| ProductId {FK} | Foreign key, product identifier. | int | No |
| UpdatedPrice | The price user proposed. | int | No |
| LastUpdatedTime | The time user proposed price. | datetime | No |

Table 4: Attribute Data Dictionary

## Sequence Diagram

### Force Parse Data

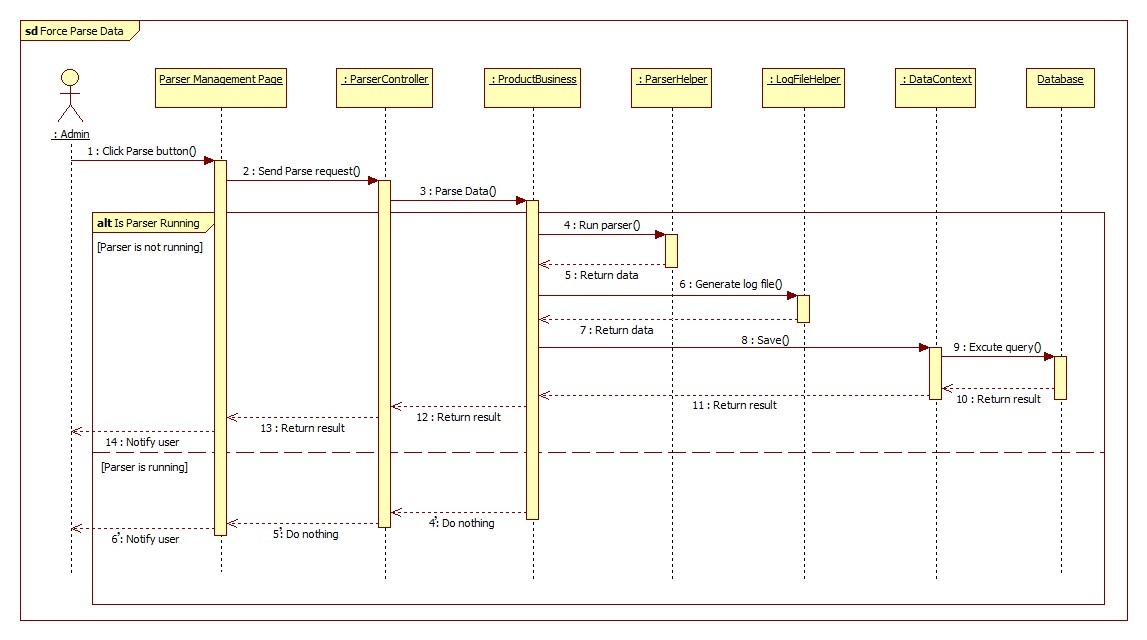


Figure 7: Force Parse Data Sequence Diagram

### Import Product



Figure 8: Import Product Sequence Diagram

### Process Error Products



Figure 9: Process Error Products Sequence Diagram

### Process Duplicated Products



Figure 10: Process Duplicated Products Sequence Diagram

### Save Correct Products



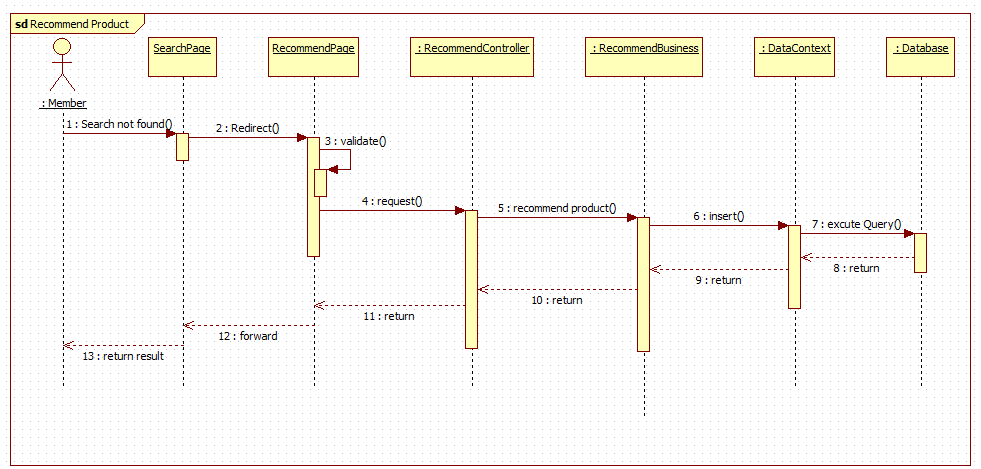
Figure 11: Save Correct Products Sequence Diagram

### Log File



Figure 12: Log File Sequence Diagram

### Recommend



#### Figure 13: Recommend Sequence Diagram

### View History

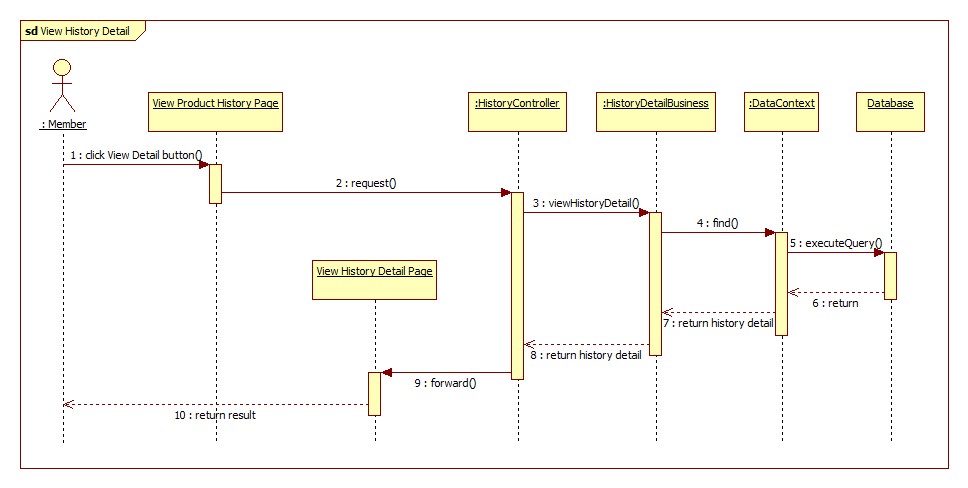


Figure 14: View History Sequence Diagram

### Search Product

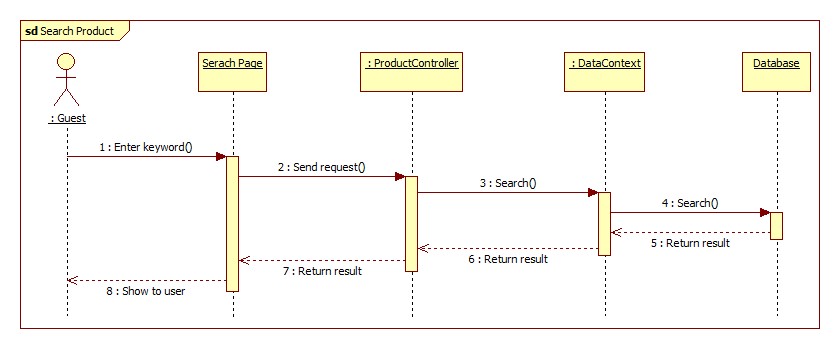


Figure 15: Search Product Sequence Diagram

### Configure System

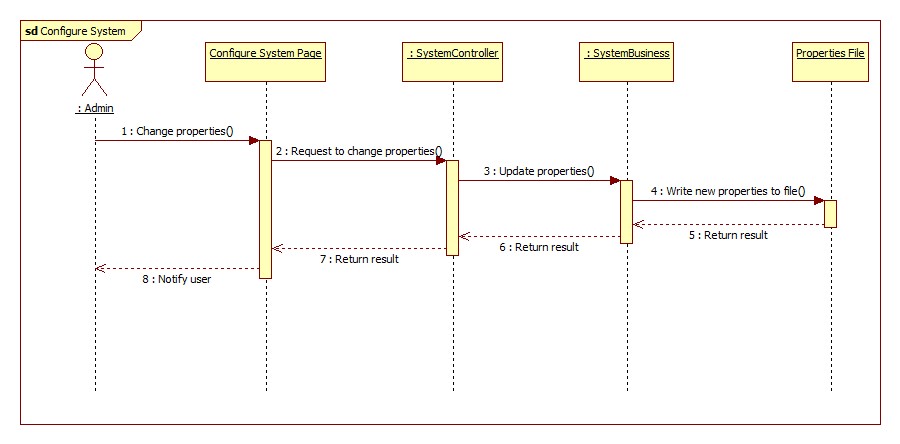


Figure 16: Configure System Sequence Diagram

### Confirm New Product



Figure 17: Confim New Product

### Training Machine



Figure 18: Training Machine

## Algorithms

### Calculating Product Point

Chờ Việt nơi ấy

### String Comparison

##### Define Problem

* Given two strings. Calculate their matching percent.

##### Requirement

* Robustness to changes of word order: two strings which contain the same words, but in a different order, should be recognised as being similar.
* Language independence: the algorithm should work not only in English, but in many different languages.

##### Solution

* If a string contains many words, break it into a list of words.
* For each word, we find out how many adjacent character pairs are contained in it.
* Create a function *pairs(s)* which returns a list of adjacent character pairs of string *s*.
* Then, we use below formula to calculate matching percent.

##### Example

Calculate the matching percent of 2 strings: France and French.

* Upper case 2 strings:
  + France FRANCE.
  + French FRENCH.
* Break string into list of adjacent character pairs:
  + FRANCE
  + FRENCH
* Calculate its matching percent.

## Demo

### Scenario

* Kịch Bản demo nè Việt

## Advantages and disadvantages

### Advantages

* Provide full of details suggest about specified product that you want to know and see what product is better than the other ones. It saves your time, because it takes a few minutes instead of several hours to compare the details (all text) and still don’t know which one is under your budget and better. You will never miss out a special, coupon or rebate. Finally, you can get real life reviews, recommend and rating from the other users.
* The system collect product form abundant sources.

### Disadvantages

* Staffs spend time for tranining.
* Not support mobie version.
* Not provide Payment Method.
* Staff must have knowledge about hardware.