**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

Smart Buy

|  |  |
| --- | --- |
| **Group 1** | |
| **Group member** | Doan Ho Anh Triet – Team Leader – SE60763  Huynh Thanh Viet – Team Member - SE60666  Dang Huu Hoang – Team Member - 60486  Tran Trung Dung – Team Member - 60236 |
| **Supervisor** | Mr. Kieu Trong Khanh |
| **Ext. Supervisor** | N/A |
| **Capstone Project code** | SmartB |

-Ho Chi Minh City, 02/2014-

*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.Kiều Trọng Khánh, 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**](#_Toc369941326)

[**List of Tables 5**](#_Toc369941327)

[**List of Figures 7**](#_Toc369941328)

[Definitions, Acronyms, and Abbreviations 9](#_Toc369941329)

[Report No.4 Software Desgin Description 10](#_Toc369941330)

[1. Design Overview 10](#_Toc369941331)

[2. System Architectural Design 11](#_Toc369941332)

[3. Component Diagram 12](#_Toc369941333)

[4. Detailed Description of Components 13](#_Toc369941334)

[4.1 Class Diagram 13](#_Toc369941335)

[4.2 Class Diagram Explanation 13](#_Toc369941336)

[4.3 Sequence Diagram 21](#_Toc369941337)

[5. Database Design 33](#_Toc369941338)

[5.1 Logical Diagram 33](#_Toc369941339)

[5.2 Data Dictionary 34](#_Toc369941340)

[6. Algorithms 40](#_Toc369941341)

[6.1 Face Detection 40](#_Toc369941342)

[6.2 Face Recognition 44](#_Toc369941343)

# List of Tables

[Table 1: Entity Data Dictionary 32](#_Toc369941379)

[Table 2: Attribute Data Dictionary 37](#_Toc369941380)

# List of Figures

[Figure 1: MVC Architecture 9](#_Toc369941449)

[Figure 2: Component Diagram 10](#_Toc369941450)

[Figure 3: Class Diagram 11](#_Toc369941451)

[Figure 4: Add Image for Singe Student Sequence Diagram 19](#_Toc369941452)

[Figure 5: Add Image for Many Students Sequence Diagram 19](#_Toc369941453)

[Figure 6: Delete Student Image Sequence Diagram 20](#_Toc369941454)

[Figure 7: Create Account Sequence Diagram 20](#_Toc369941455)

[Figure 8: Configure System Sequence Diagram 21](#_Toc369941456)

[Figure 9: Face Detection Sequence Diagram 21](#_Toc369941457)

[Figure 10: Face Recognition Sequence Diagram 22](#_Toc369941458)

[Figure 11: Auto Free Storage Space Sequence Diagram 22](#_Toc369941459)

[Figure 12: Auto Active Roll Call Sequence Diagram 23](#_Toc369941460)

[Figure 13: Add Roll Call Sequence Diagram 23](#_Toc369941461)

[Figure 14: Edit Roll Call Sequence Diagram 24](#_Toc369941462)

[Figure 15: Import Student List Sequence Diagram 24](#_Toc369941463)

[Figure 16: Export Report Sequence Diagram 25](#_Toc369941464)

[Figure 17: Add Student Sequence Diagram 25](#_Toc369941465)

[Figure 18: Add Class Sequence Diagram 26](#_Toc369941466)

[Figure 19:Edit Class Sequence Diagram 26](#_Toc369941467)

[Figure 20: Add Subject Sequence Diagram 27](#_Toc369941468)

[Figure 21: Edit Subject Sequence Diagram 27](#_Toc369941469)

[Figure 22: Take Attendance Auto Sequence Diagram 28](#_Toc369941470)

[Figure 23: Take Attendance Manual Sequence Diagram 28](#_Toc369941471)

[Figure 24: View Roll Call Detail Sequence Diagram 29](#_Toc369941472)

[Figure 25: Report Attendance by Class Sequence Diagram 29](#_Toc369941473)

[Figure 26: Check Present Rate Sequence Diagram 30](#_Toc369941474)

[Figure 27: Logical Diagram 31](#_Toc369941475)

[Figure 28: Example of Haar Future 38](#_Toc369941476)

[Figure 29: Apply Haar future to sub-window 39](#_Toc369941477)

[Figure 30: Classifier Cascade 39](#_Toc369941478)

[Figure 31: Viola-Jones method flowchart 41](#_Toc369941479)

[Figure 32: Face Recognition Process 42](#_Toc369941480)

[Figure 33: Face Recognition Algorithm Flowchart 44](#_Toc369941481)

# Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| SmartB | Smart Buy |

# Report No.4 Software Design Description

## Design Overview

* This document describes the technical and user interface design of Smart Buy System using mobile device. It includes the architectural design, the detailed design of common functions and business functions and the design of database model.
* The architectural design describes the overall architecture of the system and the architecture of each main component and subsystem.
* The detailed design describes static and dynamic structure for each component and functions. It includes class diagrams, class explanations and sequence diagrams for each use cases.
* The database design describes the relationships between entities and details of each entity.
* Document overview:
* Section 2: gives an overall description of the system architecture design.
* Section 3: gives component diagrams that describe the connection and integration of the system.
* Section 4: gives the detail design description which includes class diagram, class explanation, and sequence diagram to details the application functions.
* Section 5: describe an ERD with logical diagram.

## System Architectural Design



Figure 1: MVC Architecture

**(http://www.w3schools.com/aspnet/mvc\_intro.asp)**

**The 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.

**The View** is the parts of the application that handles the display of the data.  
Most often the views are created from the model data.

**The 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

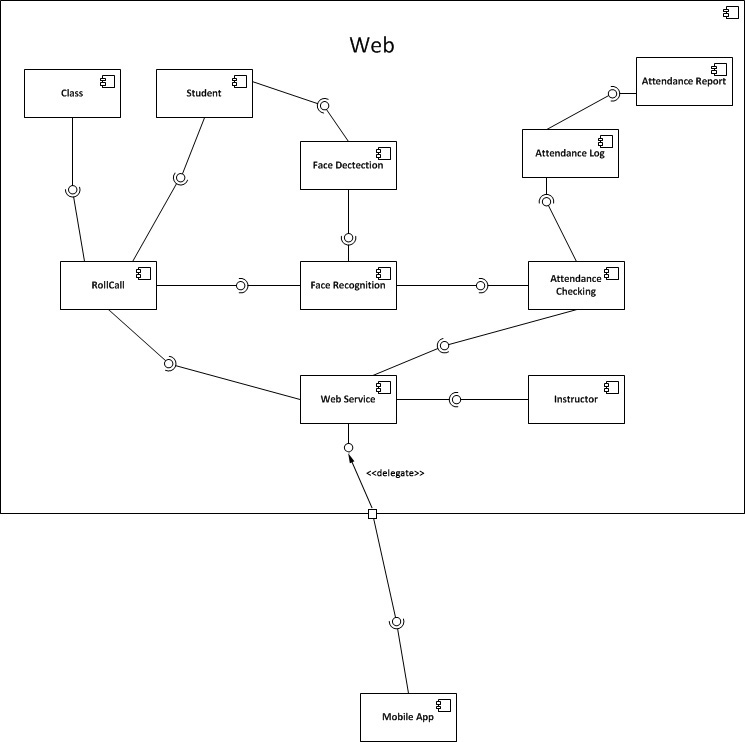


Figure 2: Component Diagram

## Detailed Description of Components

### Class Diagram

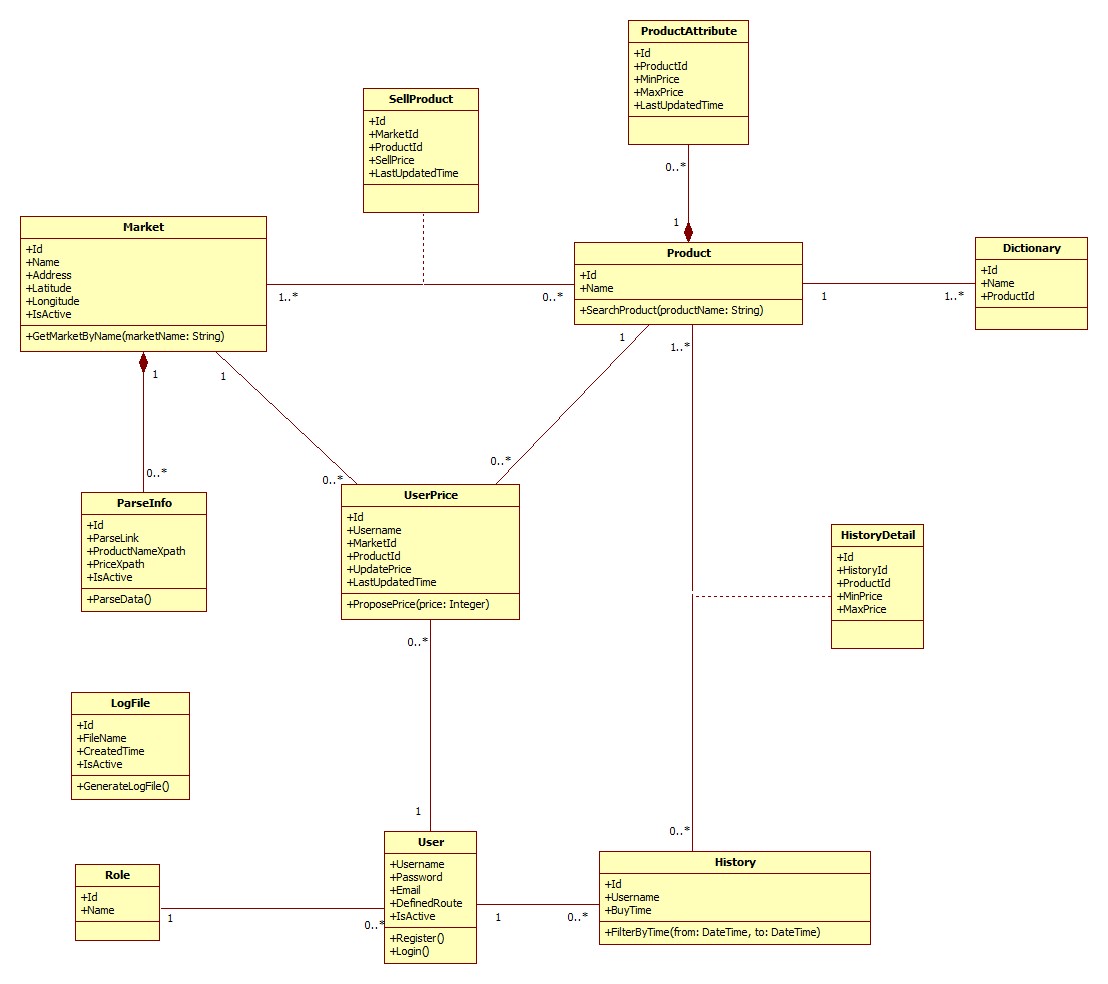


Figure 3: Class Diagram

### Class Diagram Explanation

#### Market

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each market |
| Name | String | Public | Name of market |
| Address | String | Public | Address of market |
| Longitude | Double | Public | Longitude of market |
| IsActive | Boolean | Public | Status of market |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| GetMarketByName | List of markets | Public | Find market by its name |

#### Parse Info

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each item |
| ParseLink | String | Public | Link used to parse |
| ProductNameXpath | String | Public | Xpath to get the name of products |
| PriceXpath | String | Public | Xpath to get the price of products |
| IsActive | Boolean | Public | Status of info |

Method

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

#### Sell Product

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each item |
| MarketId | Integer | Public | Id of market |
| ProductId | Integer | Public | Id of product |
| SellPrice | Integer | Public | That market sells that product at this price |
| LastUpdatedTime | DateTime | Public | The time when the price is updated. |

#### Product

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each product |
| Name | Integer | Public | Product name |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| SearchProduct | List of products | Public | Search product by its name |

#### Product Attribute

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each item |
| ProductId | Integer | Public | Id of product |
| MinPrice | Integer | Public | Min price of this product |
| MaxPrice | Integer | Public | Max price of this product |
| LastUpdatedTime | DateTime | Public | The time when this information is updated. |

#### User Price

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique identifier of each item |
| Username | String | Public | Username of member who propose price |
| MarketId | Integer | Public | Id of market |
| ProductId | Integer | Public | Id of product |
| UpdatePrice | Integer | Public | Proposed price |
| LastUpdatedTime | DateTime | Public | The time when the price is proposed. |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| ProposePrice | Void | Public | Propose price for the system |

#### User

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Username | String | Public | Username for each user |
| Password | String | Public | User’s password |
| Email | String | Public | User’s email |
| DefinedRoute | String | Public | The route which user defines |
| IsActive | Boolean | Public | Status of this account |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Register | Void | Public | For new user to register |
| Login | Boolean | Public | Used to log in the system |

#### History

Attribute

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

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| FilterByTime | List of history | Public | Filter history list by the range of time |

#### 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 |
| MinPrice | Integer | Public | Min price of that product at that time |

#### 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 |

#### Dictionary

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Public | Unique id of each item |
| Name | String | Public | The name of the product |
| ProductId | Integer | Public | The identifier of that product |

### Sequence Diagram

#### Force Parse Data

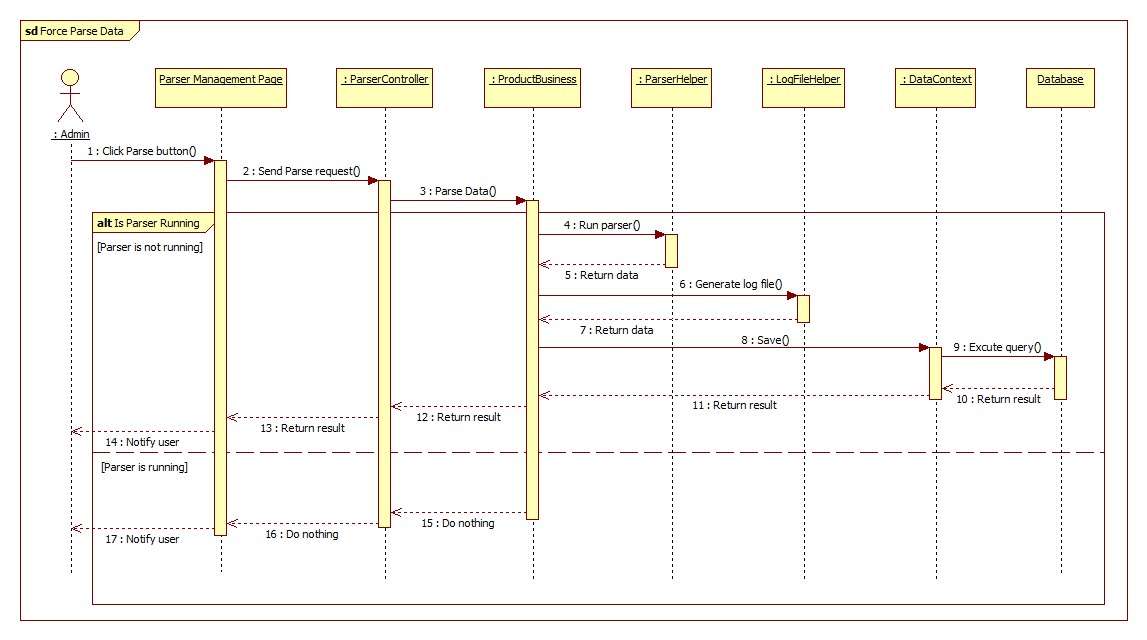


Figure 4: Force Parse Data Sequence Diagram

#### Import Excel

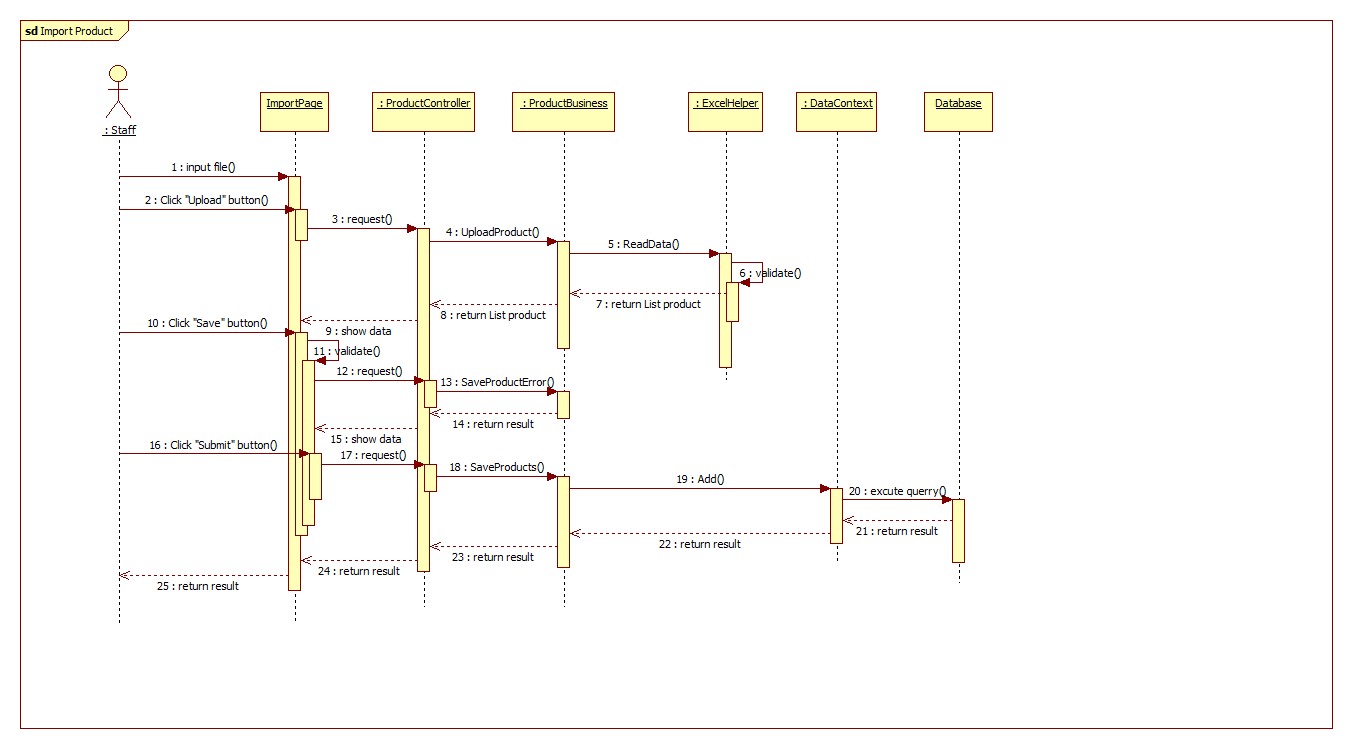


Figure 5: Import Excel Sequence Diagram

#### Propose Price

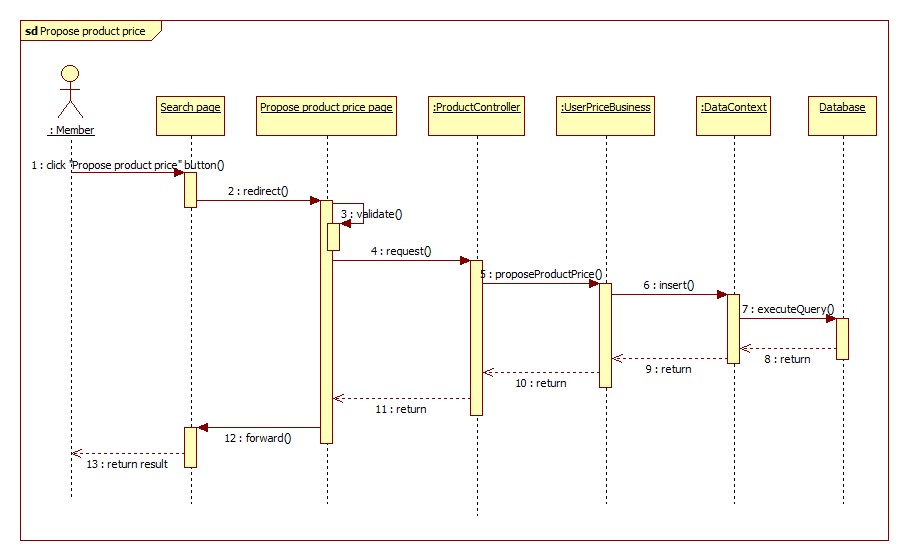


Figure 6: Propose Price Sequence Diagram

#### Save Product History

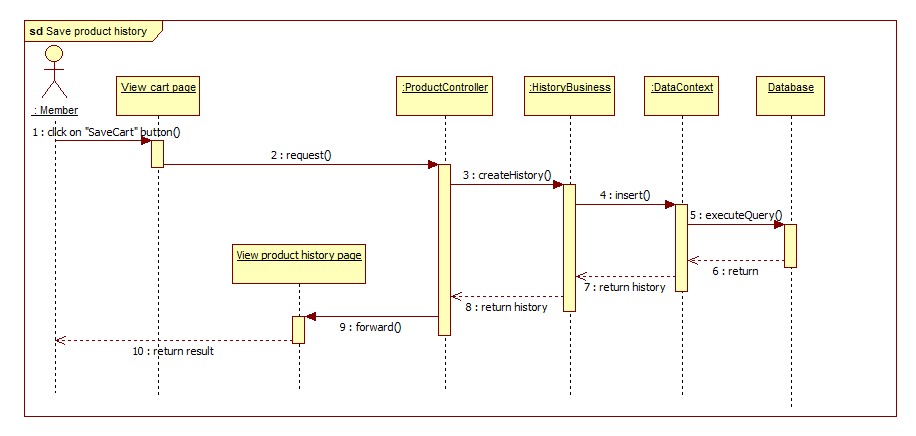


Figure 7: Save Product History Sequence Diagram

#### View Product History

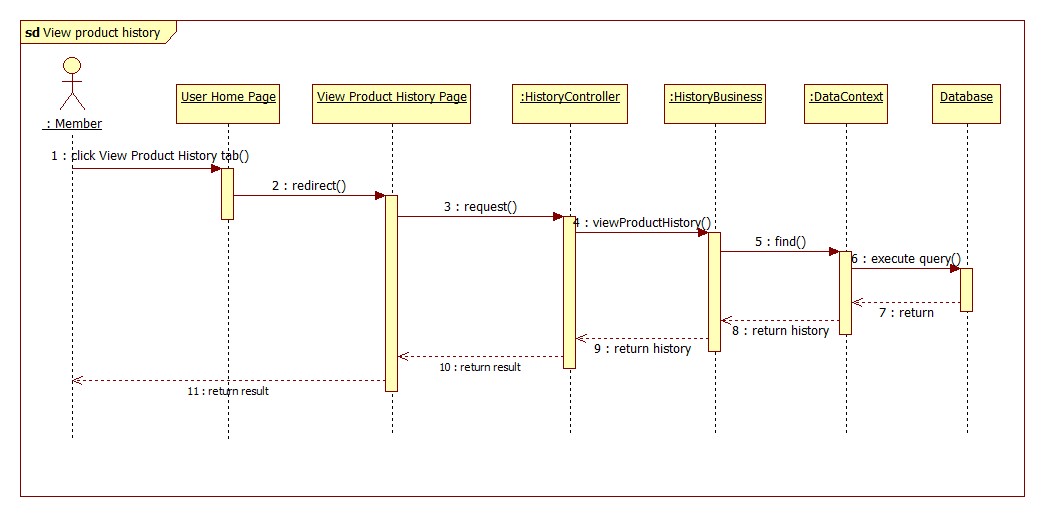


Figure 8: View Product History Sequence Diagram

#### View History Detail

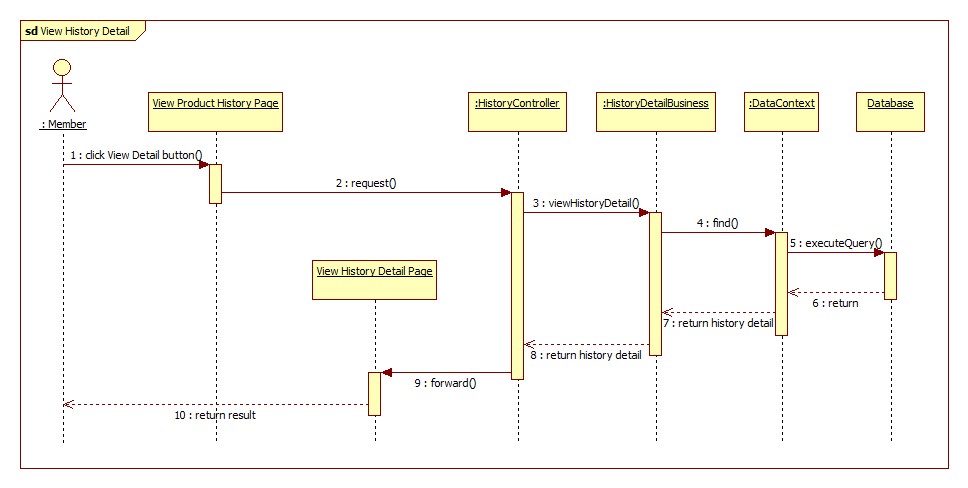


Figure 9: View History Detail Sequence Diagram

#### <Mobile> Search Product

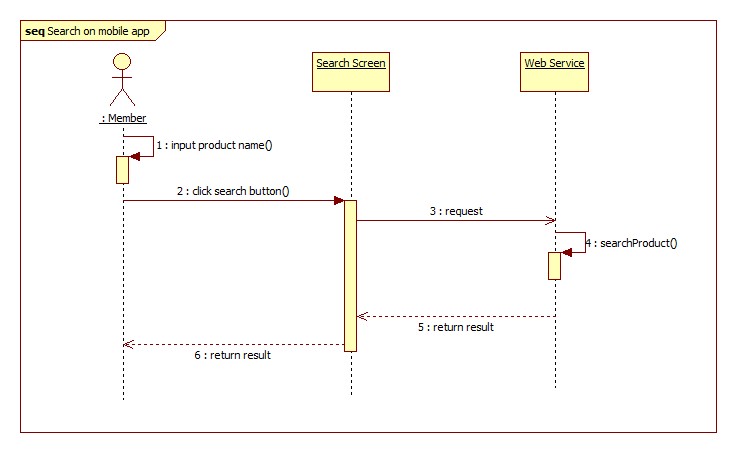


Figure 10: <Mobile> Search Product Sequence Diagram

#### Configure System

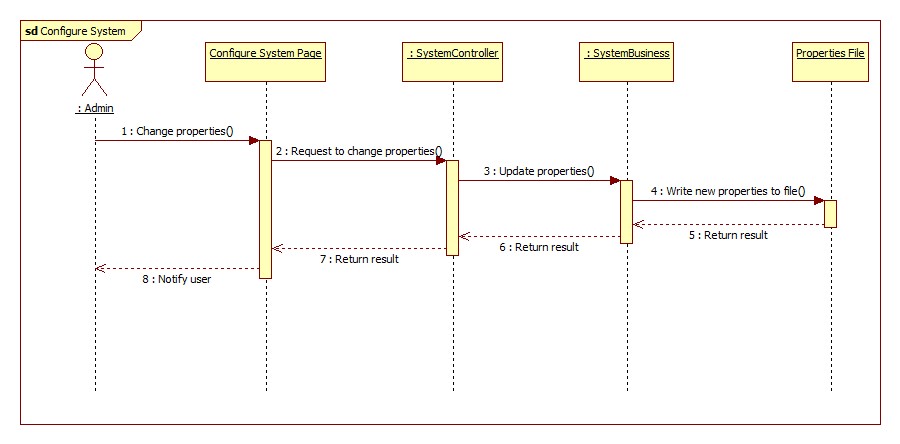


Figure 11: Configure System Sequence Diagram

#### Ask for System Suggestion

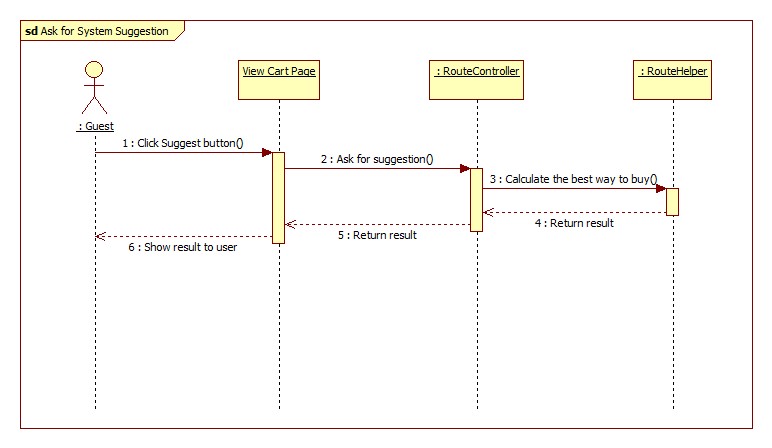


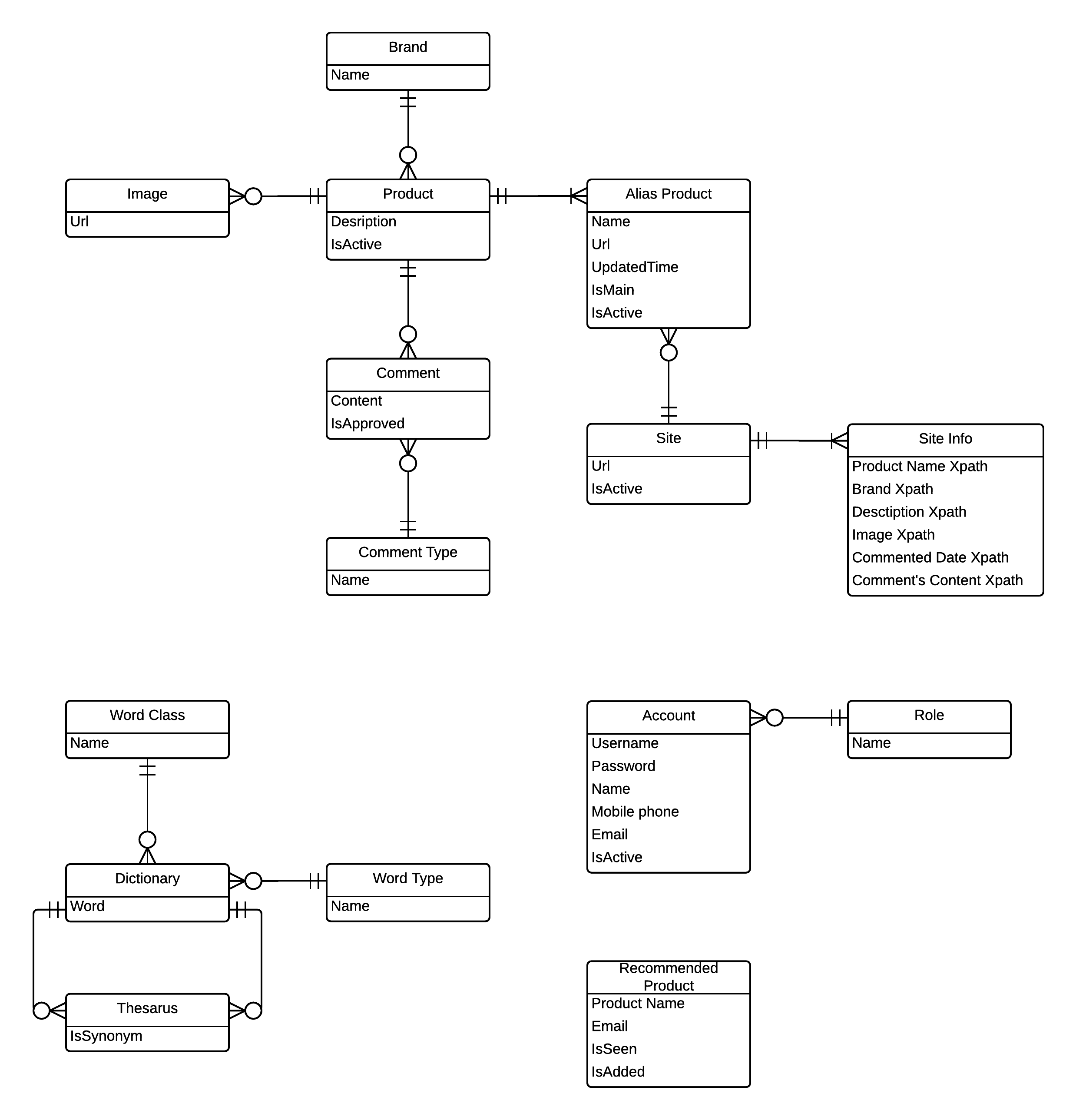
Figure 12: Ask for System Suggestion Sequence Diagram

## User Interface Design

## Database Design

### Logical Diagram

Figure 27: Logical Diagram



### Data Dictionary

|  |  |
| --- | --- |
| **Entity Data dictionary: describe content of all entities** | |
| **Entity Name** | **Description** |
| Market | Describe all markets in the system. |
| ParseInfo | Describe all data needed to parse websites. |
| SellProduct | Describe relationship between Market and Product. Markets sell products. |
| Product | Describe all products in the system. |
| ProductAttribute | Describe all attributes of a product. |
| Dictionary | Describe all possible product names in the system. |
| User | Describe all accounts in the system. Account includes: admin, staff, member, guest… |
| UserPrice | Describe the relationship between Market, User, and Product. User proposes the price of a product at a market. |
| History | Describe all buying 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. |

Table 1: Entity Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity name** | **Attributes** | **Description** | **Domain** | **Nulls** |
| Semester | SemesterID{PK}  SemesterName  BeginDate  EndDate | Uniquely identifies a semester, auto increment.  Name of semester.  The begin date of semester.  The end date of semester. | Int  15 nvar  Datetime  Datetime | No  No  No  No |
| Major | MajorID{PK}  ShortName  FullName | Uniquely identifies a major, auto increment.  Short name of major.  Full name of major. | Int  2 nvar  30 nvar | No  Yes  No |
| SubjectMajorMapping | MajorID {FK}  SubjectID {FK} | Foreign key references to table Major  Foreign key references to table Subject | Int  Int | No  No |
| Subject | SubjectID{PK}  ShortName  FullName  NumberOfSlot  NumberOfSession  IsSlotFixed  Description  IsActive  TypeID {FK} | Uniquely identifies a subject, auto increment.  Short name of subject.  Full name of subject.  Number of slot of subject (per day).  Number of session of subject.  This value shows that a slot can fixed or not.  Description of subject  This value shows that the subject is active or not.  Foreign key references to table Type | Int  10 nvar  50 nvar  Int  Int  Bit  10 char  Bit  Int | No  Yes  No  No  No  No  Yes  No  No |
| SubjectType | TypeID {PK}  TypeName | Uniquely identifies a type, auto increment.  Name of type. | Int  50 nvar | No  No |
| Class | ClassID {PK}  MajorID {FK}  ClassName  IsActive | Uniquely identifies a class, auto increment.  Foreign key references to table Major  Name of class  This value shows that the class is active or not. | Int  Int  6 char  Bit | No  No  No  No |
| Student | StudentID {PK}  ClassID {FK}  FullName  Birthdate  CitizenID  Address  Email  IsActive  UserID {FK}  StudentCode | Uniquely identifies a student, auto increment.  Foreign key references to table Class.  Full name of student.  Birthdate of student.  CitizenID of student  Address of student  Email of student.  This value shows that student is active or not.  Foreign key references to table User  Student code of student. | Int  Int  50 nvar  Datetime  8 char  100 nvar  50 nvar  Bit  Int  7 char | No  No  No  No  No  Yes  Yes  No  Yes  No |
| StudentImage | ImageID {PK}  StudentID {FK}  ImageLink | Uniquely indentifies of image, auto increment.  Foreign key references to table Student  The link of image. | Int  Int  100 var | No  No  No |
| Instructor | InstructorID {PK}  FullName  Email  Phone  IsActive  UserID {FK}  SubjectTypeID {FK} | Uniquely identifies of instructor, auto increment.  Full name of instructor.  Email of instructor.  Phone of instructor.  This value shows the instructor is active or not.  Foreign key references to table User.  Foreign key references to table SubjectType. The instructor can only teach subject with this type | Int  50 nvar  50 nvar  12 nvar  Bit  Int  Int | No  No  Yes  Yes  No  Yes  No |
| StudySession | SessionID {PK}  RollcallID {FK}  InstructorID {FK}  SessionDate  StartTime  EndTime  ClassID {FK}  Note | Uniquely identifies of session, auto increment.  Foreign key references to table Rollcall.  Foreign key references to table Instructor.  The date of session.  The time session start.  The time session end.  Foreign key references to table Class.  Note of session. | Int  Int  Int  Date  7 time  7 time  Int  50 nvar | No  No  No  No  No  No  No  Yes |
| Rollcall | RollcallID {PK}  StartTime  EndTime  BeginDate  EndDate  SubjectID {FK}  ClassID {FK}  SemesterID {FK}  Status  InstructorID {FK} | Uniquely identifies of session, auto increment.  The time roll call start.  The time roll call end.  The date roll call begins.  The date roll call end.  Foreign key references to table Subject.  Foreign key references to table Class.  Foreign key references to table Semester.  Status of roll call. (0: created. 1: active. 2: unactive)  Foreign key references to table Instructor. The main instructor of the roll call | Int  7 time  7 time  Date  Date  Int  Int  Int  Int  Int | No  No  No  No  No  No  No  No  No  No |
| StudentInRollcall | StudentID {FK}  RollcallID {FK} | Foreign key references to table Student.  Foreign key references to table Roll Call. | Int  Int | No  No |
| StudentAttendance | LogID {FK}  StudentID {FK}  IsPresent  Note | Foreign key references to table AttendanceLog.  Foreign key references to table Student.  This value shows that student is present or not.  Note of student attendance. | Int  Int  Bit  50 nvar | No  No  No  Yes |
| AttendanceLog | LogID {PK}  RollcallID {FK}  LogDate  TypeID {FK} | Uniquely identifies of attendance, auto increment.  Foreign key references to table Rollcall.  The date of log.  Foreign key references to table LogType. | Int  Int  Date  Int | No  No  No  No |
| LogImage | ImageID {PK}  LogID {FK}  ImageLink | Uniquely identifies of image, auto increment.  Foreign key references to table AttendanceLog.  The link of image. | Int  Int  100 var | No  No  No |
| LogType | TypeID {PK}  TypeName | Uniquely identifies of log type, auto increment.  Name of type. | Int  10 nvar | No  Yes |
| Staff | StaffID {PK}  FullName  Email  Phone  IsActive  UserID {FK} | Uniquely identifies of staff, auto increment.  Full name of staff.  Email of staff.  Phone of staff.  This value shows that staff is active or not.  Foreign key references to table User. | Int  50 nvar  50 nvar  12 nvar  Bit  Int | No  No  Yes  Yes  No  Yes |
| User | UserID {PK}  UserName  Password  RoleID {FK}  IsActive | Uniquely identifies of user, auto increment.  Username of user.  Password of user.  Foreign key references to table Role.  This value shows that user is active or not. | Int  30 nvar  30 nvar  Int  Bit | No  No  No  No  No |
| Role | RoleID {PK}  RoleName | Uniquely identifies of role, auto increment.  Name of role. | Int  10 nvar | No  No |

Table 2: Attribute Data Dictionary

## Algorithms

### Dynamic Programming

#### Definition

In [mathematics](http://en.wikipedia.org/wiki/Mathematics), [computer science](http://en.wikipedia.org/wiki/Computer_science), [economics](http://en.wikipedia.org/wiki/Economics), and [bioinformatics](http://en.wikipedia.org/wiki/Bioinformatics), dynamic programming is a method for solving complex problems by breaking them down into simpler sub-problems. It is applicable to problems exhibiting the properties of [overlapping sub-problems](http://en.wikipedia.org/wiki/Overlapping_subproblem) and [optimal substructure](http://en.wikipedia.org/wiki/Optimal_substructure). When applicable, the method takes far less time than naive methods that don't take advantage of the sub-problem overlap (like [depth-first search](http://en.wikipedia.org/wiki/Depth-first_search)).

References: <http://en.wikipedia.org/wiki/Dynamic_programming>

#### Define Problem

In our system, there is a function called “Ask for System Suggestion”. This function gives users the best way to buy their list of products based on their cart and the markets around their defined route. Here we can describe this problem as below:

**Given a two-dimensional array named Price, size means market sells product at price. Find a way to buy all products at the minimum cost.**

**The output should be in this format**:

* **Buy <product name> at <market name>.**
* **Buy <product name> at <market name>.**
* **Buy <product name> at <market name>.**
* **…**

#### Solution

To solve this problem, we should follow these steps:

* Call **M** is the number of total products; **N** is the number of total markets.
* Create a constant named **Penalty**. It stands for the cost of moving between two markets.
* Create a two-dimensional array named **Total** size . is the sum of money user have to pay when the buying process is at item in market .
* Create a one-dimensional array named **TraceY**, size **N**. means we buy the product before product *i* at market .
* Initialize all elements in **Total** with a very big number.
* Set the first row of **Total** the same as the first row of **Price**.
* Use this regression formula to calculate the rest of **Total**:
* The minimum value at the final row of **Total** is the answer.
* Trace back to find the complete answer.

#### Complexity

* To traverse through all elements in **Total** array, we go through steps.
* With each element, we traverse its previous row. Each row has elements.
* In total, the complexity of this algorithm is

#### Flowchart

