**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

**Laptop Reviews**

|  |  |
| --- | --- |
| **Group 1** | |
| **Group member** | Vo Thi Minh Chau – Team Leader – SE60931  Nguyen Van Hon – Team Member - 60390 (Dropped out)  Dinh Huu Toan – Team Member - SE60871  Nguyen Manh Khuong – Team Member - 60455 |
| **Supervisor** | Mr. Kieu Trong Khanh |
| **Ext. Supervisor** | N/A |
| **Capstone Project code** | LRA |

-Ho Chi Minh City, 05/2015-

*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 iv](#_Toc427442808)

[List of Tables vi](#_Toc427442809)

[List of Figures vii](#_Toc427442810)

[Definitions, Acronyms, and Abbreviations viii](#_Toc427442811)

[A. Project Management Plan 1](#_Toc427442812)

[1. Project Definition 1](#_Toc427442813)

[1.1 Project Information 1](#_Toc427442814)

[1.2 Problem Abstract 1](#_Toc427442815)

[1.3 Project Overview 1](#_Toc427442816)

[2. Project Organization 3](#_Toc427442817)

[2.1 Software Process Model 3](#_Toc427442818)

[2.2 Roles and responsibilities 3](#_Toc427442819)

[2.3 Tools and Techniques 4](#_Toc427442820)

[3. Project Management Plan 6](#_Toc427442821)

[Software Development Life Cycle 6](#_Toc427442822)

[3.1 System Overview Use Case 8](#_Toc427442823)

[4. Software System Attribute 10](#_Toc427442824)

[4.1 Usability 10](#_Toc427442825)

[4.2 Reliability 10](#_Toc427442826)

[4.3 Availability 10](#_Toc427442827)

[4.4 Security 10](#_Toc427442828)

[4.5 Maintainability 10](#_Toc427442829)

[4.6 Portability 10](#_Toc427442830)

[4.7 Performance 10](#_Toc427442831)

[5. Conceptual Diagram 10](#_Toc427442832)

[Data Dictionary 11](#_Toc427442833)

[B. Software Requirement Specification 13](#_Toc427442834)

[1. User Requirement Specification 13](#_Toc427442835)

[1.1 Guest Requirement 13](#_Toc427442836)

[1.2 Member Requirement 13](#_Toc427442837)

[1.3 Staff Requirement 13](#_Toc427442838)

[1.4 Admin Requirement 13](#_Toc427442839)

[1.5 System Requirement 13](#_Toc427442840)

[2. System Requirement Specification 14](#_Toc427442841)

[System Overview Use Case 15](#_Toc427442842)

[3. Main Flows 16](#_Toc427442843)

[4. Entity Relationship – Conceptual 17](#_Toc427442844)

[C. Software Design Description (SDD) 18](#_Toc427442845)

[1. Design Overview 18](#_Toc427442846)

[2. System Architectural Design 18](#_Toc427442847)

[3. Component Diagram 18](#_Toc427442848)

[4. Detailed Description of Components 19](#_Toc427442849)

[4.1 Class Diagram 19](#_Toc427442850)

[4.2 Class Diagram Explanation 19](#_Toc427442851)

[4.3 Algorithms 19](#_Toc427442852)

# List of Tables

[Table 1: Hardware Requirement for Server 2](#_Toc427442792)

[Table 2: Hardware Requirement for Mobile Application 2](#_Toc427442793)

[Table 3: Roles and Responsibility Details 4](#_Toc427442794)

[Table 4: Software Development Life Cycle Detail 7](#_Toc427442795)

[Table 5: Entity Data dictionary - Describe content of all entities 12](#_Toc427442796)

# List of Figures

[Figure 1: The Scrum Process 3](#_Toc427442797)

[Figure 2: System Overview Use Case 9](#_Toc427442798)

[Figure 3: Conceptual Diagram 11](#_Toc427442799)

[Figure 4: System Overview Use Case 15](#_Toc427442800)

[Figure 5: Main Flows 16](#_Toc427442801)

[Figure 6: Entity Relationship Diagram 17](#_Toc427442802)

[Figure 7: Component Diagram 18](#_Toc427442803)

[Figure 8: Class Diagram 19](#_Toc427442804)

# Definitions, Acronyms, and Abbreviations

|  |  |  |
| --- | --- | --- |
| **No.** | **Abbreviation & Acronym** | **Definition** |
| 1 | LRA | Laptop Reviews |
| 2 | OS | Operation System |
| 3 | Admin | Administrator |
| 4 | API | Application Programming Interface |
| 5 | HTTP | Hyper Text Transfer Protocol |

# Project Management Plan

## Project Definition

### Project Information

* Project name: **Laptop Review**
* Project Code: **LRA**

### Problem Abstract

Today, when people want to buy a new laptop, they usually search its information though technology websites. The information like laptop’s specifications, prices, outfit pictures are typically found, but lack of reviews from customer who bought that laptop. To solve that problem, we propose a solution which can help people to archive it. We will gather and classify the reviews from trusted websites so that customers can have the best advices and easily make decisions. How can we know whether the reviews are positive or negative? Our system will, therefore, help classify the reviews automatically, show them to our customers in an appropriate way.

### Project Overview

#### Current Situation

In Vietnam, people tend to choose laptops base on what they hear from sellers at electronic markets or what they read on some technical forums, websites. These activities have limitations. Sellers’ advices may be not accurate, some reviews may be non-sense and are not classified. Moreover, it will take lots of time for people to come to electronic markets to have laptops’ information or read reviews on many forums, websites. After searching on Google, we find this page: [www.buydig.com](http://www.buydig.com). They offer classified reviews for laptops but not all laptops. Therefore, customers will be upset when they can’t find what they need. Our solution will do a better job. We will gather and classify the reviews from trusted websites so that customers can make the best decision. Moreover, customers can claim for reviews for laptops which they can’t see on our system and get notification when those laptops’ information is updated.

#### The Proposed System

The system is intended for use by those with a smart phone or a laptop/computer with Internet connection. The system will have the following functions:

##### Web Application

* Admins can manage accounts.
* System can parse product, classify review and store to database daily or on requests.
* Staff can check feedback from user, manual update dictionary.
* Users can search laptop’s information, leave feedback

##### Mobile Application

Users can search laptops, write comments or add favourite laptops.

#### Boundaries of the System

* The system can be used by every people with a smart phone or a laptop/computer with Internet connection.
* The language of the system is English.
* The complete product includes:
* The website, for staff and user.
* All the process document involved.

#### Development Environment

##### Hardware Requirement

**For Server**

|  |  |  |
| --- | --- | --- |
| Windows | Minimum Requirements | Recommended |
| Internet Connection | Cable, Wifi (4 Mbps) | Cable, Wifi (8 Mbps) |
| Operating System | XP, Vista, 7, 8 | XP, Vista, 7, 8 |
| Computer Processor | Intel® Core 2 Duo | Intel® Core(TM) i5 CPU , M 460 @ 2.53GHz |
| Computer Memory | 1GB RAM | 3GB or more |

Table 1: Hardware Requirement for Server

**For Mobile Application**

|  |  |  |
| --- | --- | --- |
| Android | Minimum Requirements | Recommended |
| Internet Connection | Wifi (4 Mbps) | Wifi (8 Mbps) |
| Operating System | 4.1 Jelly Bean | 4.3 Jelly Bean or later |
| Computer Processor | 1.0 GHz | 1.2 GHz or more |
| Computer Memory | 512 MB RAM | 1GB or more |

Table 2: Hardware Requirement for Mobile Application

##### Software requirements

* Microsoft Windows 8.1: operating system and platform for development.
* SQL Server 2008 Express R2: used to create and manage the database for system.
* StarUML: used to create models and diagrams.
* Skype: used for communication and meeting.
* Visual Studio 2013: used to implement website and web service.
* Github.com & TortoiseSVN: used for source control.

## Project Organization

### Software Process Model

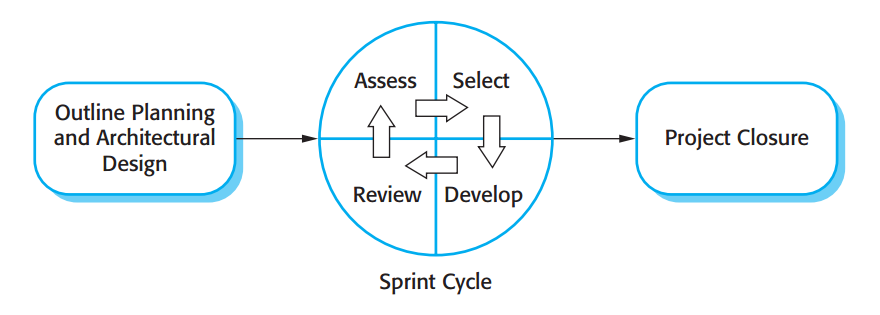


Figure 1: The Scrum Process

(Sommerville, Software Engineering, 9th Edition, 2011, Figure 3.8 page 73)

### Roles and responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Full name** | **Role in Group** | **Responsibilities** |
| **1** | Kieu Trong Khanh | Scrum Master | * Arrange daily meetings * Track the backlog of work to be done * Record decisions * Measure progress against the backlog * Communicates with customers and management outside of the team |
| **2** | Vo Thi Minh Chau | Scrum Team Member | * Design database * Clarify requirements * Prepare documents * GUI Design * Create test plan * Code * Test |
| **3** | Nguyen Van Hon | Scrum Team Member | * Design database * Clarify requirements * Prepare documents * GUI Design * Create test plan * Code * Test |
| **4** | Dinh Huu Toan | Scrum Team Member | * Design database * Clarify requirements * Prepare documents * GUI Design * Create test plan * Code * Test |
| **5** | Tran Manh Khuong | Scrum Team Member | * Design database * Clarify requirements * Prepare documents * GUI Design * Create test plan * Code * Test |

Table 3: Roles and Responsibility Details

### Tools and Techniques

* Front-end technologies: HTML5, CSS3, JavaScript, jQuery 1.10, AJAX, SignalR 2.0.
* Back-end: Website: ASP.NET MVC5 + Entity Framework 5.
* Web Server: Microsoft IIS 7.
* Database Management System: MS SQL Server 2008 Express R2.
* - Android Developer Tools: Eclipse

## Project Management Plan

### Software Development Life Cycle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phase**  **/Iteration** | **Description** | **Deliverables** | **Resource needed** | **Dependencies and Constrains** | **Risks** |
| **Outline Planning and Architectural Design** | - Study similar existing systems.  -Identify and clarify requirements for the system in general. | - Introduction of proposed system.  - Project task plan.  - Software requirement specification.  - Prototypes. | 20 man-days | N/A | - Missing  requirement  - Unclear  scope of  project  - Lack of  member share  of understand |
| **Sprint Cycle – Parser Analysis** | - Analyze websites to parse data.  - Choose a library for parsing HTML content. | - Websites which will be parsed date.  - Library for parsing HTML content. | 10 man-days | N/A | - Lack of experience. |
| **Sprint Cycle – Training Machine Function** | -Teach the system how to synchronize products’ names. | N/A | 20 man-days | Depends on “Data management”. | - Not have a clear understanding about business process. |
| **Sprint Cycle – Dictionary Management** | - Input data manually.  - Import data from excel files.  - Find synonyms and antonyms form dictionary websites. | - Dictionary management service. | 30 man-days | N/A | - Lack of experience.  - The dictionary is not variety. |
| **Sprint Cycle – Parser Management** | - Manage parser. | - Parser management system. | 20 man-days | Depend on “Sprint Cycle – Parser Analysis”. | - Lack of experience.  - Not have a clear understanding about business process. |
| **Sprint Cycle – Main User’s Functions** | - Parse data.  - Build algorithm to analyze comments then classify them into 3 groups: positive, negative and neutral.  - Member (staff and admin) can manage profile. | - Collected data.  - Classified comment system.  - Main user’s functions on web. | 60 man-days | Depend on “Sprint Cycle – Parser Analysis” and “Sprint Cycle - Dictionary management”. | - The implemented algorithm is not the best.  - Lack of test data.  - Lack of experience on analyzing sentence’s meaning. |
| **Sprint Cycle – User Account**  **Management** | - Manage user accounts in the system. | - Account management system. | 5 man-days | N/A | - Lack of experience.  - Not have a clear understanding about business process. |
| **Project Closure** | - Documentation. | - Installation guide.  - User manual. | 3 man-days | Depend on “Sprint Cycle” | - Lack of experience. |

**Table 4: Software Development Life Cycle Detail**

Software Requirement Specification

### System Overview Use Case

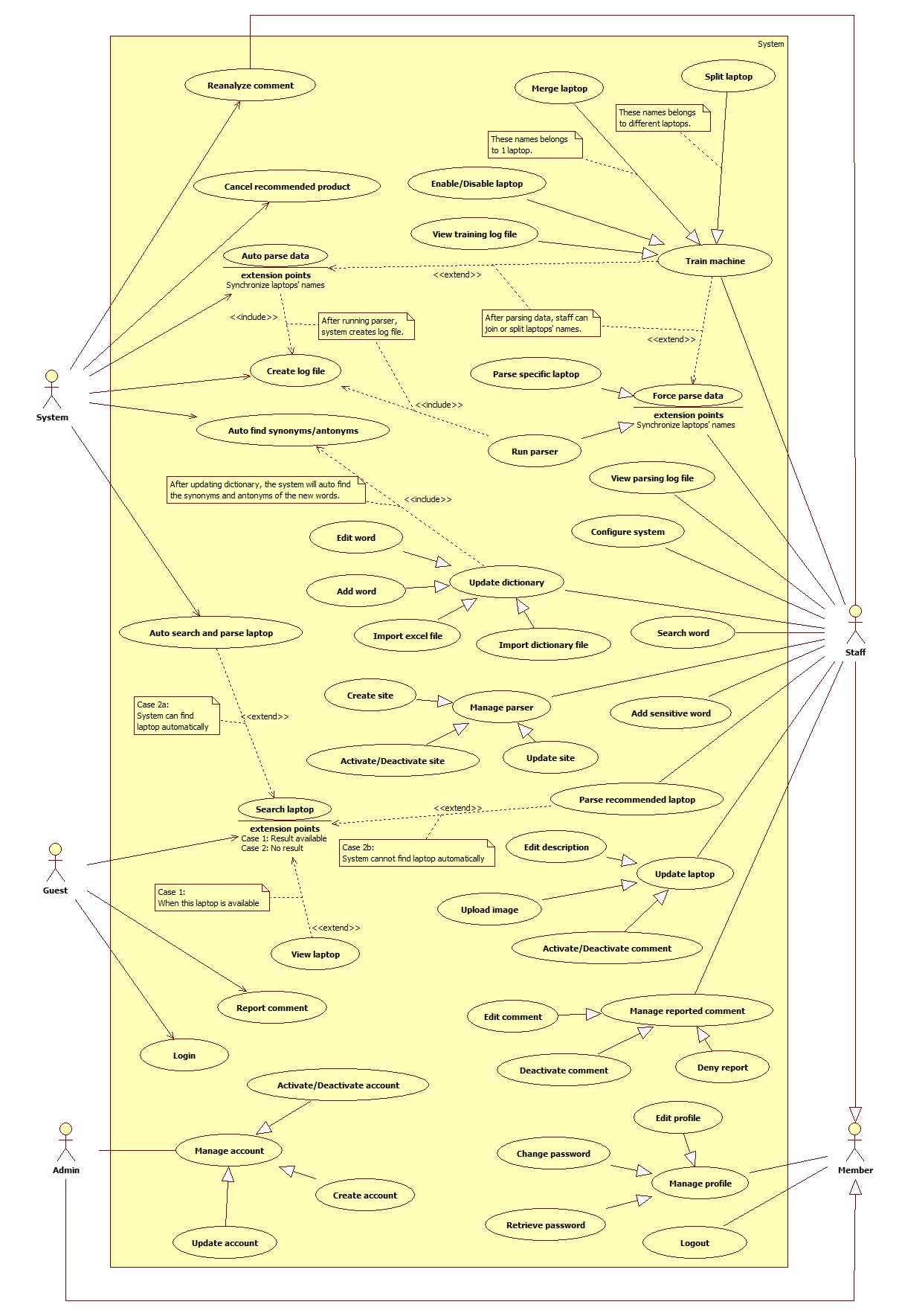


Figure 2: System Overview Use Case

## Software System Attribute

### Usability

#### Graphic User Interface

* All the texts, labels and should be written in English.

#### Usability

* Admin, staff should need less than one week of training to be productive with the system.
* Guest and member can use the system easily without training.

#### Installation

* The system must be easy to deploy. Customer can deploy successfully and learn to configure, maintain the system within one day of training.
* The attached manual guide must be clear. User can read and do themselves without developer’s help.

### Reliability

* The product should be updated every day.
* The parser should be updated every day.
* If the product is already existed, system will notify for user.
* The system can analyze almost comments, classify them base on dictionary with above 80% accuracy.

### Availability

* N/A.

### Security

* Privacy: Each role of user has a specific permission to interact with system.
* System always checks authorization and authentication before doing anything.
* Only admin can grant permission to staff.

### Maintainability

* Dictionary data is easy to change and update.

### Portability

* The web application can be used anywhere, where has Internet Connection, by Smart phones or PC.

### Performance

* Server responses in 3 seconds for normal progresses, in 10 seconds for complex progresses.
* The page resources are lightly to reduce loading time.

## Conceptual Diagram

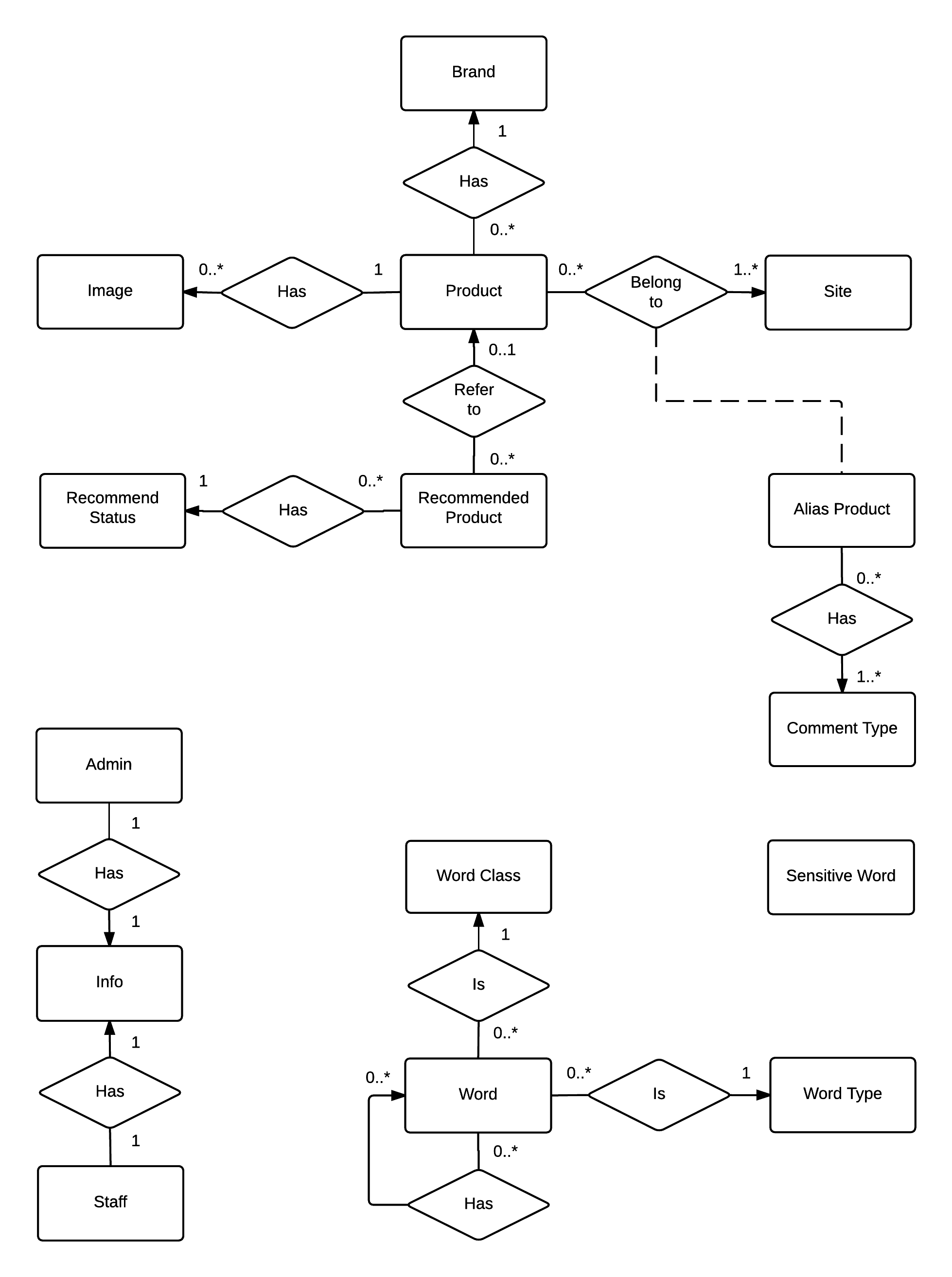


Figure 3: Conceptual Diagram

### Data Dictionary

|  |  |
| --- | --- |
| **Entity Data dictionary: describe content of all entities** | |
| **Entity Name** | **Description** |
| Image | Describe all image link that product is included. |
| Product | Describe all products in the system. |
| Recommend Product | Describe the product that user recommend for system. |
| Recommend Status | Describe the status of user recommend. |
| Brand | Describe all brands the products are belonged. |
| Alias Product | Describe all alias products in the system. |
| Comment Type | Describe all types of comments. |
| Site | Describe all site parse data. |
| Word | Describe all words |
| Word Type | Describe all types of words in Dictionary. |
| Word Class | Describe all classes of words in Dictionary |
| Staff | Describe all information of staff. |
| Admin | Describe all information of admin. |
| Info | Describe all information of user. |

Table 5: Entity Data dictionary - Describe content of all entities

# Software Requirement Specification

## User Requirement Specification

### Guest Requirement

Guests is a person who doesn’t have access to the system. Guest can use some functions in the system. These are some functions guest can use:

Guests are normal users who don’t have access to the system. Guests can use almost functions but the systems functions. Here are things guests can do with our website:

* Search products.
* View products details with all information.
* Report unsuitable comment.
* Recommend not available products and ask for notification when they are available.

### Member Requirement

Member is an authorised user of the system who has some functions relating their account, such as:

* Edit profile.
* Change password
* Retrieve password.

### Staff Requirement

Staff is a person whose work is maintaining the system. Staff will be able to do all Member’s functions and following additional these:

* Update dictionary manually or through importing Excel file and dictionary file.
* Modify parser.
* Train the system so that it can handle some tasks such as managing duplicated products.
* Configure the system
* Force parsing data.
* Manage products.
* Parse recommend product.
* Handle reported comment.

### Admin Requirement

Admin is the one who manage all accounts in the system. Beside the Member’s function, Admin can be able to:

* Create account.
* Edit account.
* Activate/Deactivate account.

### System Requirement

System is also an actor help run the website. System will handle these works:

* Auto parse the data
* Auto find synonyms and antonyms
* Parse recommended product if the result can be found automatically

## System Requirement Specification

### System Overview Use Case

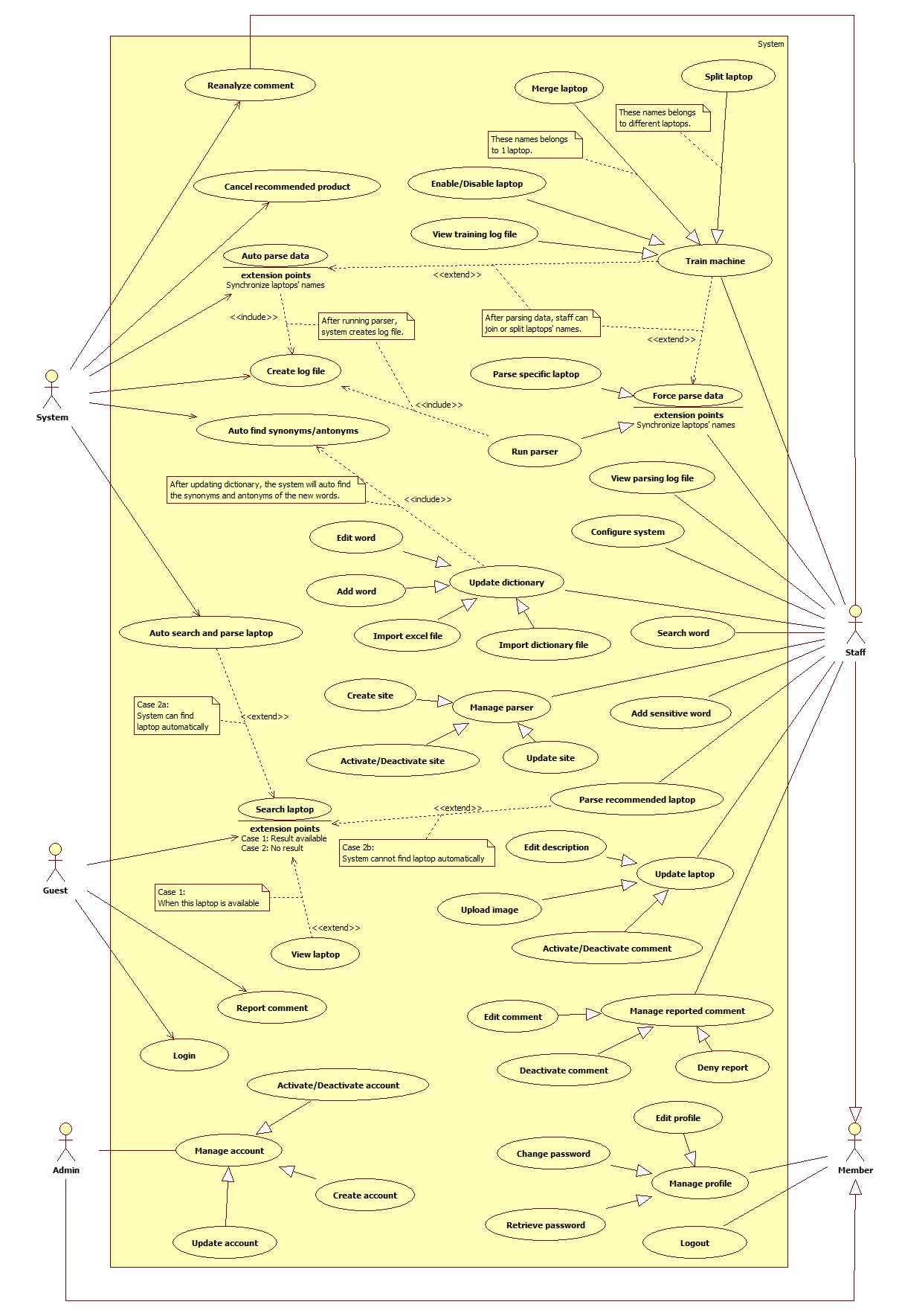


Figure 4: System Overview Use Case

## Main Flows

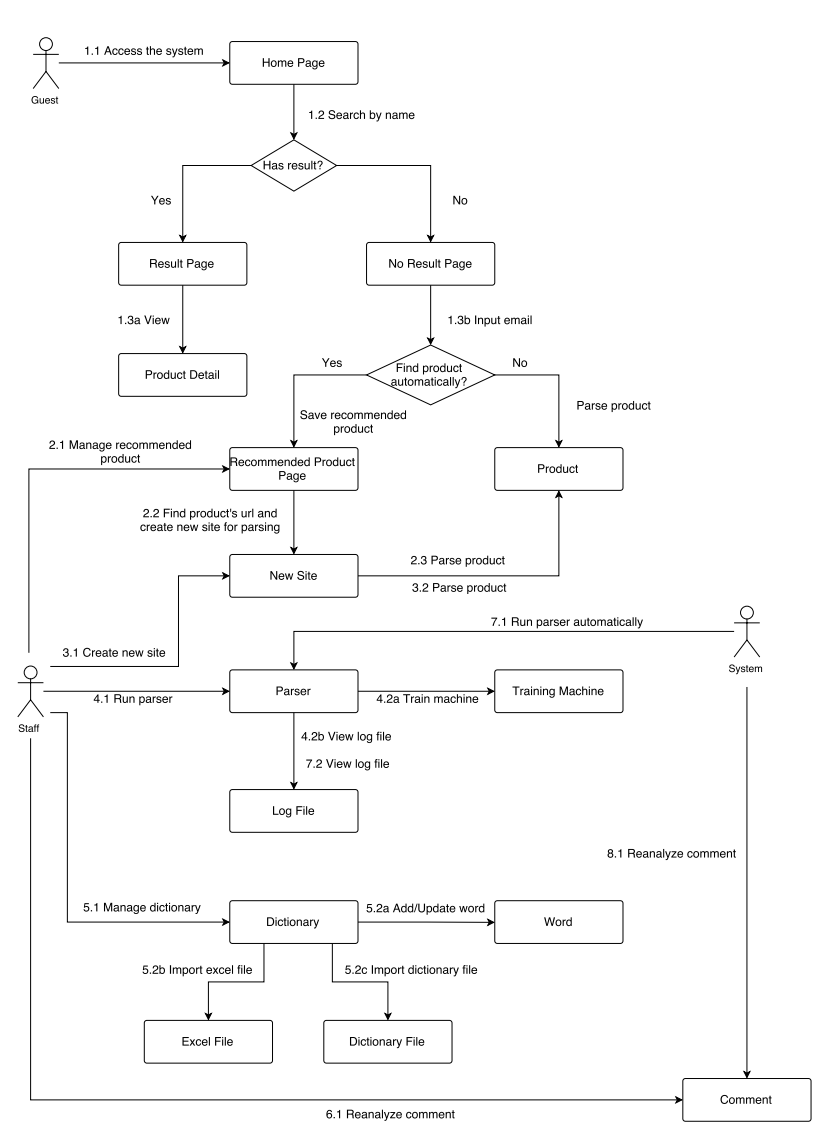


Figure 5: Main Flows

## Entity Relationship – Conceptual

Entity Relationship Diagram (ERD)

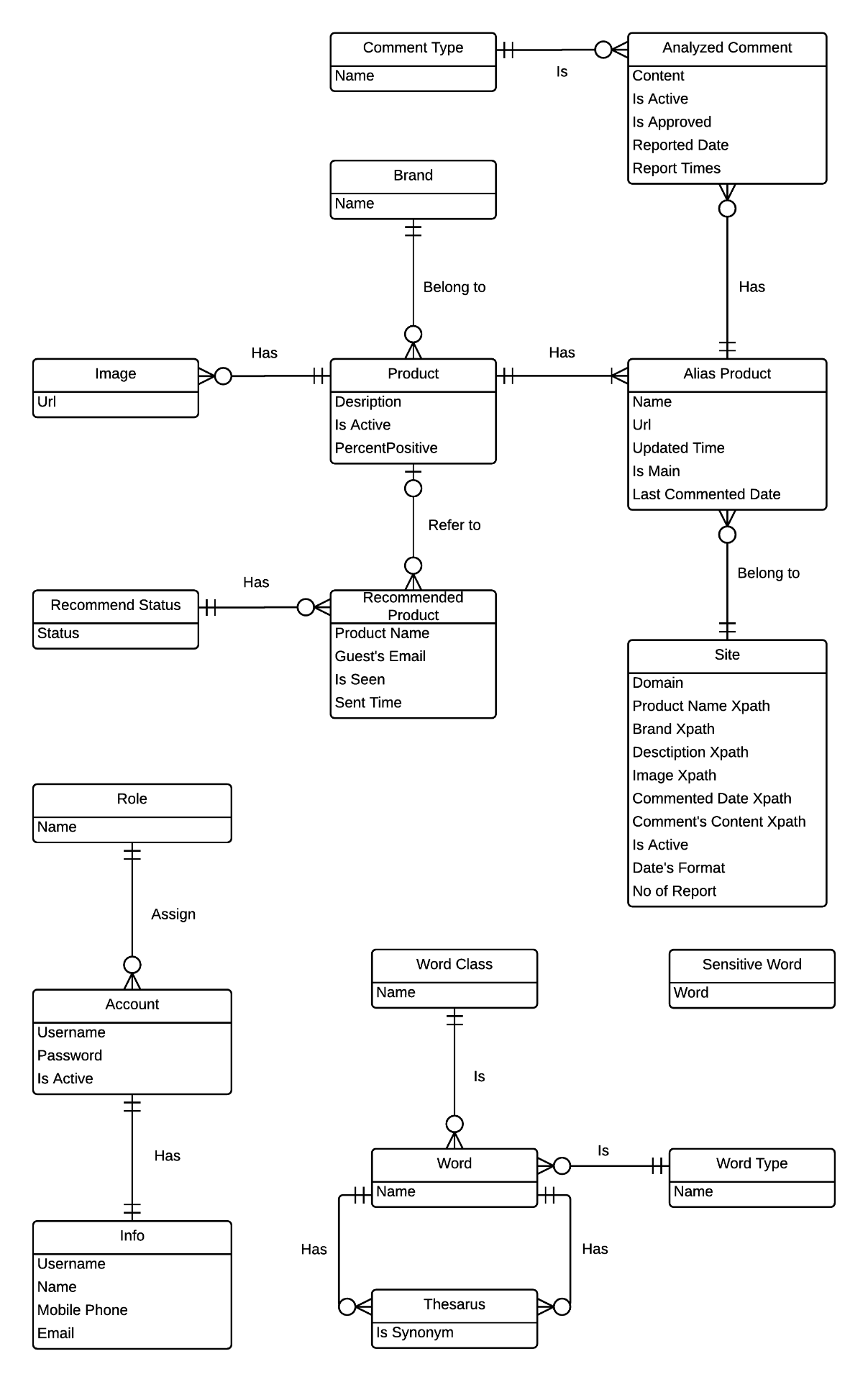


Figure 6: Entity Relationship Diagram

# Software Design Description (SDD)

## Design Overview

* This document describes the technical and user interface design LRA System using web. 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 entities.
* 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

## Component Diagram

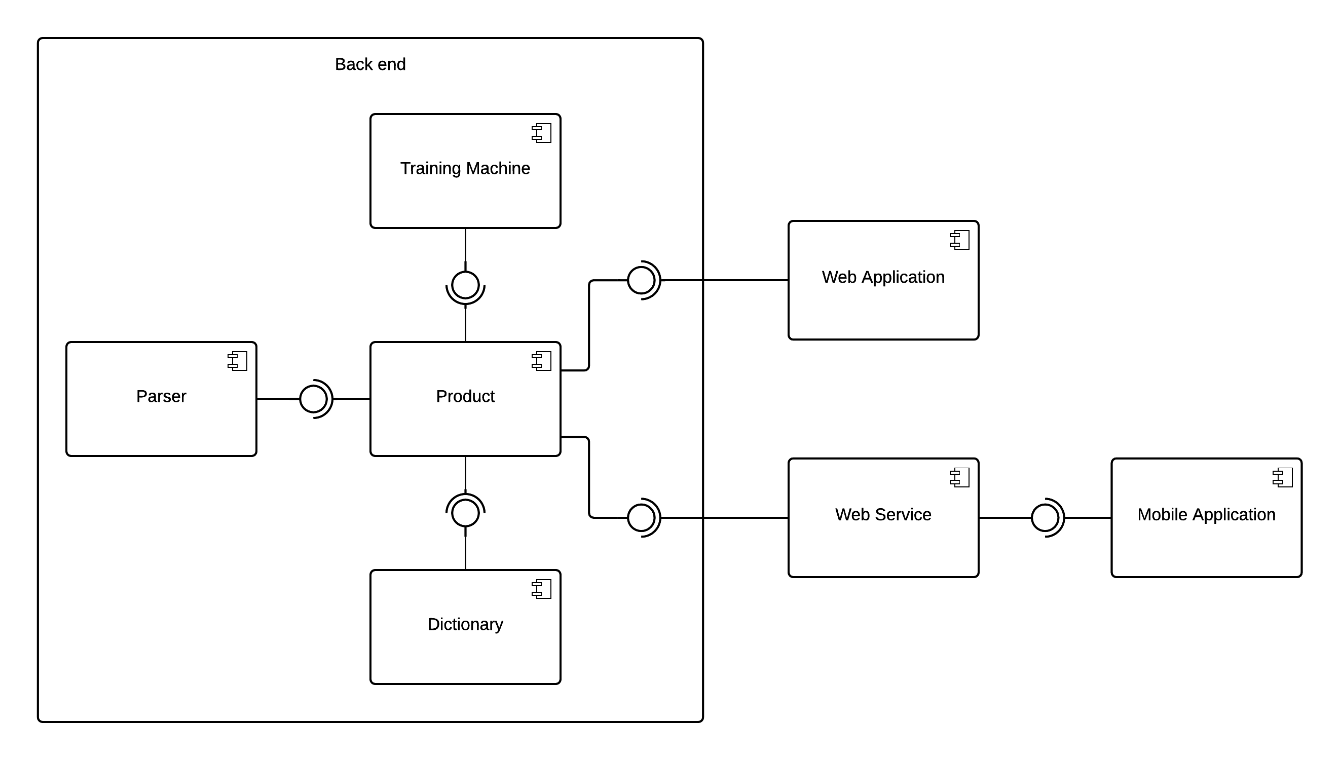


Figure 7: Component Diagram

## Detailed Description of Components

### Class Diagram

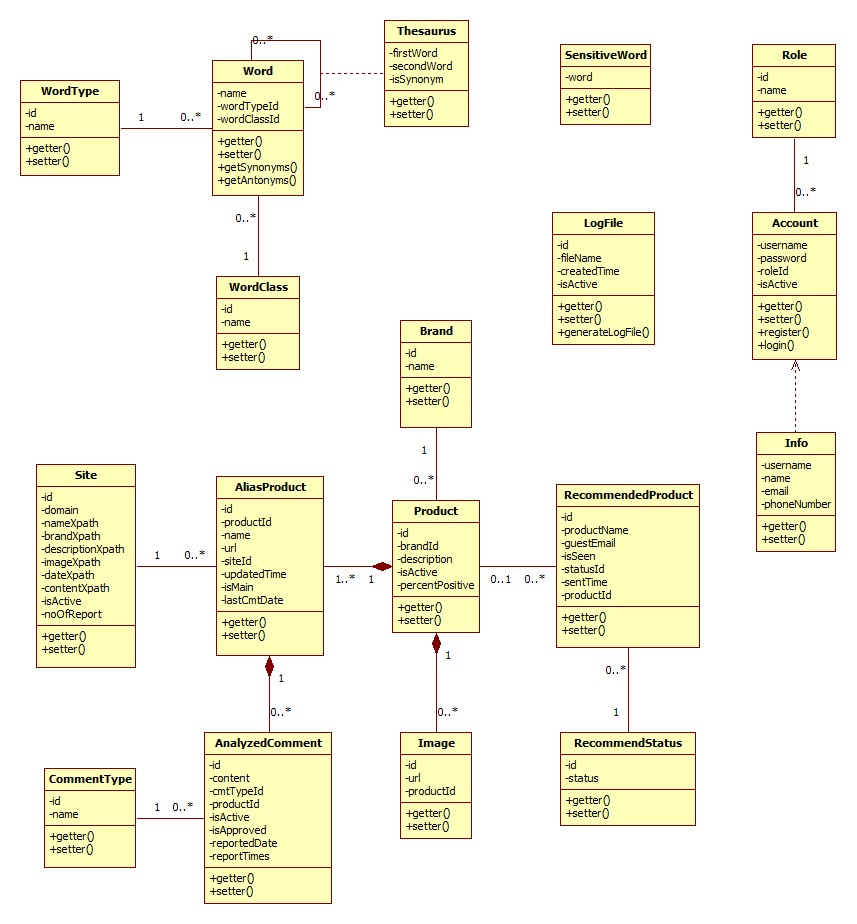


Figure 8: Class Diagram

### Class Diagram Explanation

#### Site

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | integer | Public | Unique identifier of each site |
| url | string | Public | Link to the site |
| nameXpath | string | Public | Xpath that defines product’s name location in the website |
| brandXpath | string | Public | Xpath that defines product’s brand location in the website |
| descriptionXpath | string | Public | Xpath that defines product’s description location in the website |
| imageXpath | string | Public | Xpath that defines product’s image location in the website |
| dateXpath | string | Public | Xpath that defines product’s submitted date location in the website |
| contentXpath | string | Public | Xpath that defines product’s content location in the website |
| siteId | integer | Public | Id of site |
| isActive | boolean | Public | Status of site |
| noOfReport | integer | Public | Number of report |

#### Alias Product

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | integer | Public | Unique identifier of each alias product |
| productId | integer | Public | Id of product |
| name | string | Public | Name of alias product |
| url | string | Public | Link to website contains alias product |
| siteId | int | Public | Id of site |
| updatedTime | datetime | Public | Day that alias product is updated |
| isMain | boolean | Public | Use to check for main product |
| lastCmtDate | datetime | Public | Last date of comments |

#### Product

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | integer | Public | Unique identifier of each product |
| brandId | integer | Public | Id of brand |
| description | string | Public | Description of product |
| isActive | boolean | Public | Status of product |
| isReported | int | Public | State how many times the product is reported |

#### Image

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | integer | Public | Unique identifier of each image |
| url | string | Public | Link to website contains image |
| productId | integer | Public | Id of product |

#### Analyzed Comment

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | integer | Public | Unique identifier of each comment |
| content | string | Public | Content of comment |
| cmtTypeId | integer | Public | Id of comment type |
| isActive | boolean | Public | Status of comment |
| isApproved | boolean | Private |  |
| reportedDate | datetime | Public | The day the comment was reported |
| reportTimes | int | Public | State how many times the comment is reported |

#### Comment Type

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | integer | Public | Unique identifier of each comment type |
| name | string | Public | Name of comment type |

#### Brand

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | integer | Public | Unique identifier of each brand |
| name | string | Public | Name of Brand |

#### Recommended Product

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | integer | Public | Unique identifier of each recommended product |
| productName | string | Public | Name of recommended product |
| guestEmail | string | Public | Email of guest |
| isSeen | Boolean | Public | Indicate whether staff have seen the request or not |
| statusId | integer |  | Id of status |
| isAdded | Boolean | Public | Indicate whether staff have executed request or not |
| sentTime | datetime | Public | Day the recommended product is submitted |
| productId | integer | Public | Id of Product |

#### Recommended Status

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | integer | Public | Unique identifier of each user recommend |
| status | Boolean | Public | Status of Recommend |

#### Role

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | integer | Public | Unique identifier of each role |
| name | string | Public | Name of comment role |

#### Account

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| username | string | Public | Unique username of each account |
| password | string | Public | Password used to access each account |
| roleId | integer | Public | Id of role |
| isActive | Boolean | Public | Status of account |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| register | Boolean | Public | Register new account |
| login | Boolean | Public | Login to the system |

#### Info

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| username | string | Public | Unique username of each account |
| name | string | Public | Name of account owner |
| email | string | Public | Email of account owner |
| phoneNumber | string | Public | Phone number of account owner |

#### Word

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | integer | Public | Unique identifier of each word |
| name | string | Public | Name of word |
| wordTypeId | integer | Public | Id of word type |
| wordClassId | integer | Public | Id of word class |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| getSynonyms | Dictionary | Public | Get word’s synonyms |
| getAntonyms | Dictionary | Public | Get word’s antonyms |

#### WordType

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | integer | Public | Unique identifier of each word type |
| name | string | Public | Name of word type |

#### Thesaurus

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| firstWord | integer | Public | Id of word which role is main word |
| secondWord | integer | Public | Id of word which role is synonym or antonym of main word |
| isSynonym | Boolean | Public | Indicate whether the relationship of 2 words is synonym or not |

#### WordClass

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | integer | Public | Unique identifier of each word class |
| name | string | Public | Name of word class |

#### Sensitive Word

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| word | string | Public | name of sensitive word |

#### Log File

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | integer | Public | Unique identifier of each log file |
| filename | string | Public | Name of log file |
| createdTime | datetime | Public | Date the log file is created |
| isActive | Boolean | Public | Status of log file |

Method

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

### Algorithms

#### Parse Data

##### Define Problem

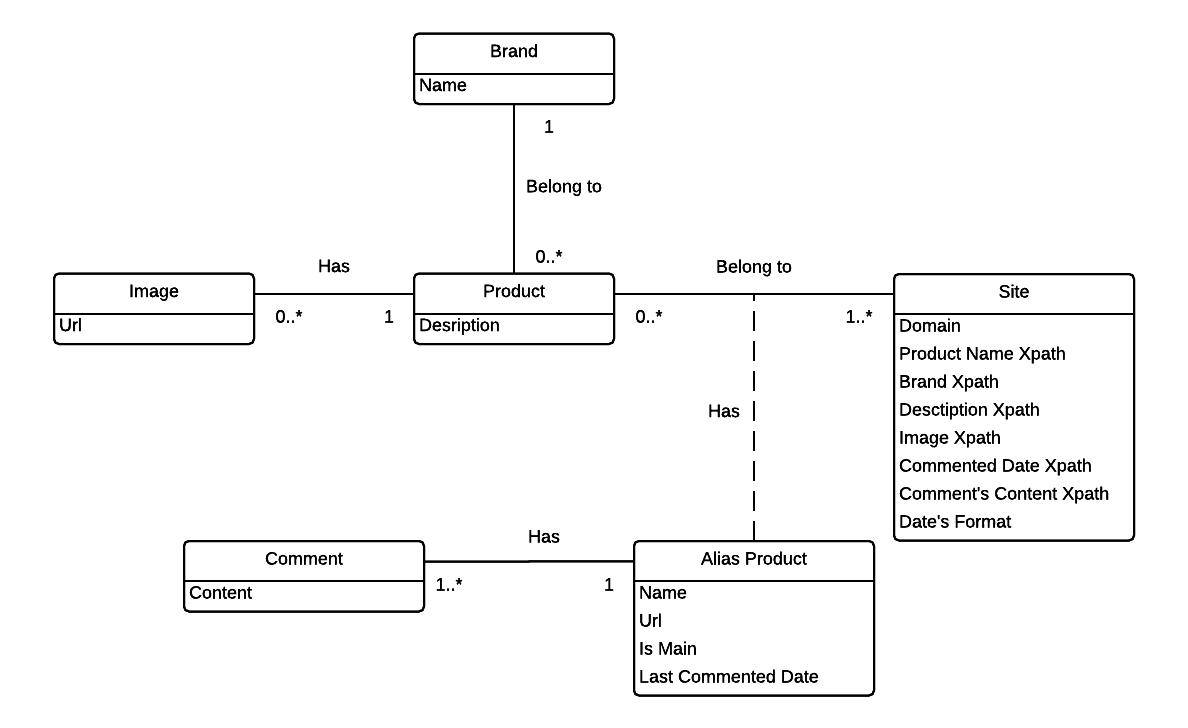
Given a laptop website then parse laptops’ specification and comments.

##### Requirement

This website must have users’ comments.

##### Solution

Our system will parse data from 3 websites: <http://www.engadget.com/>, <http://www.bestbuy.com/> and <http://www.walmart.com/>. With each product, we need to collect name and all its comments. However, to make the source more diversity, we collect these extra information, such as: brand, description and images. We also save the last comment’s date of in order to get only new comment for the next parsing time of this product. After researching from many websites, we realise that a laptop can appear in many websites with different names. We have these entities:



There are many libraries supporting parsing data and the 2 most popular are HtmlAgilityPack and Selenium. After researching 2 libraries, we decide to choose HtmlAgilityPack because during parsing process, Selenium have to create a simulator web browser to load the website’s content but HtmlAgilityPack do not have. This make improve parsing speed too much.

We intend that our system will be served to people who do not need to know what xPath is, so we develop a wizard supporting user to get xPath of anything they want. The solution is:

Step 1: From clicked node

Step 2: Get parent node

Step 3: Stop when meeting “HTML” or “ID” node

Step 4: Take xPath

Step 5: Get data

After parsing 1 product, we analyse all comments and classify them into 3 types: positive, neutral and negative comments (refer algorithm 7.2). After getting all information of this product, we compare it with existed product in database to detect if this is a duplicated product (refer algorithm 7.4). With new product, it will be inserted to database. Otherwise, duplicated product will be added to xml file and staff will handle it.

#### Analyze Comment

##### Define Problem

Given a sentence then the system will check whether that sentence has positive or negative or neutral meaning

##### Requirement

All sentence must have correct grammar. They must also have sufficient length. Moreover, all sentences that contains impolite or meaningless words will not be counted.

##### Solution

* Manually prepare 10 lists of lower-cased words by reading first 100 comments:
* A list contains all words which meanings are totally Pros
* A list contains all adjectives and adverbs which meanings are Pros (these words’ positive meaning is not as strong as words in “Totally Pros” list)
* A list contains all adjectives and adverbs which meanings are Neutral
* A list contains all adjectives and adverbs which meanings are Cons
* A list contains all adjectives and adverbs which meanings are totally Cons (these words’ negative meaning is not as strong as words in “Totally Cons” list)
* A list contains all nouns and verbs which meanings are Pros (these words’ positive meaning is not as strong as words in “Totally Pros” list)
* A list contains all nouns and verbs which meanings are Neutral
* A list contains all nouns and verbs which meanings are Cons (these words’ negative meaning is not as strong as words in “Totally Cons” list)
* A list contains all nouns and verbs which meanings are totally Cons
* A list of negative words such as not, no, do not, does not …
* Lower case the whole sentence and break it into a list of words, then lower case all the words.
* With a list of words, we will check how many words of that list belong to the 10 lists above, then we divide into these cases:
* Case 1: List of words contains word(s) which belong to “Totally Pros” word list: In this case, the sentence will be Positive sentence.
* Case 2: List of words contains word(s) which belong to “Totally Pros” word list but it also contains word(s) which belong to “Negative” word list: In this case, the sentence will be Negative sentence.
* Case 3: We will check the adjectives and adverbs fist. So if list of words contains adjectives, adverbs and belongs to adjectives/adverbs’ “Pros”, “Cons” or “Neutral” lists, we will have these sub-cases:
* If there are more “Pros” words than “Cons” words 🡪 The sentence is positive (1)
* If there are more “Cons” words than “Pros” words 🡪 The sentence is negative (2)
* With those 2 above sub-cases, if there are words belong to “Negative” list, then the sentence will be negative with sub-case (1) and positive with sub-case (2)
* If the sentence contains no adjectives/adverbs that belongs to “Pros”, “Cons” and has words belong to “Neutral”, that sentence is neutral. If the sentence has same number of “Pros” and “Cons” adjectives/adverbs and has no “Neutral” adjectives/adverbs, we will check in Case 4.
* If there is no “Pros”, “Cons” and “Neutral” adjectives/adverbs in that sentence, we will check in Case 4.
* Case 4: After checking for adjectives, adverbs, we will check verbs and nouns in that sentence. We have these sub-cases
* If there are more “Pros” words than “Cons” words 🡪 The sentence is positive (1)
* If there are more “Cons” words than “Pros” words 🡪 The sentence is negative (2)
* With those 2 above sub-cases, if there are words belong to “Negative” list, then the sentence will be negative with sub-case (1) and positive with sub-case (2)
* If the sentence contains no verbs/nouns that belongs to “Pros”, “Cons” and has words belong to “Neutral”, that sentence is neutral. Similarly, if the numbers of verbs/nouns belongs to “Pros” and “Cons” are the same, that sentence is Neutral
* If there is no “Pros”, “Cons” and “Neutral” verbs in that sentence, it will be unidentified and will be decided later by staff

##### Example

Giving the sentence: “*This Mac is fast, and combined with Mavericks I am now getting some great battery life.”*

* Assume that we already have “Totally Pros” words list which contains “great”.
* Lower case the whole sentence:

+ This Mac is fast, and combined with Mavericks I am now getting some great battery life.🡪 this mac is fast, and combined with mavericks i am now getting some great battery life.

* Split sentence into list words:

+ this mac is fast, and combined with mavericks i am now getting some great battery life 🡪 {this, mac, is, fast, and, combined, with, mavericks, i, am, now, getting, some, great, battery, life}

* We will check for “Totally Pros” words first. In this case, we have word “great”. This word belongs to “Totally Pro” list, so this sentence is Positive.

#### String Comparison

##### Define Problem

Give two strings. Calculate their matching percent.

##### Requirement

* A robustness to changes of word order: two strings which contain the same words, but in a different order, should be recognized 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: MacBook Air 2015 and MacBook Air 2015 Retina.

* Upper case 2 strings:
  + MacBook Air 2015 MACBOOK AIR 2015.
  + MacBook Air 2015 Retina MACBOOK AIR 2015 RETINA
* Break string into list of adjacent character pairs:
  + MACBOOK AIR 2015
  + MACBOOK AIR 2015 RETINA
* Calculate its matching percent.

#### Detect Duplicated Product

##### Define Problem

Give new product. Detect if it is a duplicated product.

##### Requirement

The product has been defined brand.

##### Solution

During implementation process, we discussed and improved the algorithm more times in order to improve the exactness. Here is our solution.

* First solution:
* Step 1: Using the String comparison in 7.3 to compare the new product’s name with each existed product’s name in the database. But the current formula to calculate matching percent has a little difference to the one in 7.3. The formula is:

If the result is above 40%, add these existed product to a list.

* Step 2: If the list is empty, this new product is not a duplicated product. Otherwise, choose the existed product which has the biggest comparison result in the list and the new product is duplicated product with it.

With this solution, we found 9 duplicated products when parsing 50 products.

* Second solution:
* Be improved from the 1st solution.
* At step 1, only compare the new product’s name with each existed product’s name in the database which has the same brand with new product.
* With this solution, we found 7 duplicated products when parsing 50 products – less 22% duplicated products than the 1st solution.
* Third solution:
  + - Be improved from the 2nd solution.
    - Use the different formula to compare 2 products’ names.

And the comparison result is the minimum of 2 results from above formulas.

* With this solution, we found 2 duplicated products when parsing 50 products – less 71% duplicated products than the 2nd solution.

Finally, we decided to choose the 3rd solution to detect duplicated product.