

**Elephant Learn**

**A sentimental analysis system for Thai stock market**

Software Design Document

By

Pakarat Matmarurat 592115022

Chamnol Yin 592115519

Department of Software Engineering

College of Arts, Media and Technology

Chiang Mai University

Project Advisor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr.Pree Thiengburanathum

**Document History**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **History** | **Version** | **Status** | **Date** | **Viewable** | **Editable** | **Responsible** |
| Add Chapter I   * Purpose * [Intended Audience and Reading Suggestion](bookmark://_Toc13594728)s * [Project Scope](bookmark://_Toc13594730) * Acronyms   Add Chapter II  Add Chapter III   * Class Diagram * Class Diagram Description * Sequence Diagram   Add Chapter IV   * Web application UI | Elephant  Learn-SDD\_V1.0.docx | Draft | June 21st,2019 | PT | CY, PM | CY, PM |
| Update Chapter III   * Class Diagram * Class Diagram Description * Sequence Diagram | Elephant  Learn-SDD\_V1.0.docx | Draft | July 20th,2019 | PT | CY, PM | CY, PM |

\*CY = Mr. Chamnol Yin,

\*PM = Miss Pakarat Matmarurat,

\*PT = Dr. Pree Thiengburanathum

**Table of contents**

[**Chapter I | Introduction** 5](#_Toc15388153)

[**1.1 Purpose** 5](#_Toc15388154)

[**1.2 Intended Audience and Reading Suggestions** 5](#_Toc15388155)

[**1.2.1 Intended Audience** 5](#_Toc15388156)

[**1.3 Project Scope** 5](#_Toc15388157)

[**1.4 Acronyms and Definition** 6](#_Toc15388158)

[**1.4.1 Acronyms** 6](#_Toc15388159)

[**1.4.2 Definition** 6](#_Toc15388160)

[**Chapter II | System Architecture** 8](#_Toc15388161)

[**Chapter III | Detailed Design** 9](#_Toc15388162)

[**3.1 Class Diagram** 9](#_Toc15388163)

[**3.2 Class Diagram Description** 11](#_Toc15388164)

[**3.2.1 Class name: CD-01: Spider** 11](#_Toc15388165)

[**3.2.2 Class name: CD-02: Post** 12](#_Toc15388166)

[**3.2.3 Class name: CD-03: PostComment** 13](#_Toc15388167)

[**3.2.4 Class name: CD-04: PostSpider** 13](#_Toc15388168)

[**3.2.5 Class name: CD-05: CommentSpider** 14](#_Toc15388169)

[**3.2.6 Class name: CD-06: Main** 14](#_Toc15388170)

[**3.2.7 Class name: CD-07: TextCleaning** 15](#_Toc15388171)

[**3.2.8 Class name: CD-08: Tokenization** 16](#_Toc15388172)

[**3.2.9 Class name: CD-09: Spell** 16](#_Toc15388173)

[**3.2.10 Class name: CD-10: POSTagging** 16](#_Toc15388174)

[**3.2.11 Class name: CD-11: BagOfWord** 17](#_Toc15388175)

[**3.2.12 Class name: CD-12: Sorting** 17](#_Toc15388176)

[**3.2.13 Class name: CD-13: StopWord** 18](#_Toc15388177)

[**3.2.14 Class name: CD-14: Stemming** 18](#_Toc15388178)

[**3.2.15 Class name: CD-15: Sentiment** 18](#_Toc15388179)

[**3.2.16 Class name: CD-16: Statistic** 19](#_Toc15388180)

[**3.2.17 Class name: CD-17: Result** 19](#_Toc15388181)

[**3.2.18 Class name: CD-18: FileRead** 20](#_Toc15388182)

[**3.2.19 Class name: CD-19: FirebaseHelper** 20](#_Toc15388183)

[**3.3 Sequence Diagram** 21](#_Toc15388184)

[**3.3.1 SD-01: Admin can view the retrieved comments from Pantip blog.** 21](#_Toc15388185)

[**3.3.2 SD-02: Admin can view the result of sentimental analysis of each comment in the topic.** 21](#_Toc15388186)

[**3.3.3 SD-03: Admin can view the statistic of information, which is the number of users, date, and time of the comments.** 22](#_Toc15388187)

[**Chapter IV | User Interface Design** 23](#_Toc15388188)

[**4.1 User Interface (UI)** 23](#_Toc15388189)

[**4.1.1 UI-01: Home page** 23](#_Toc15388190)

[**4.1.2 UI-02: View Result page** 24](#_Toc15388191)

[**4.1.3 UI-03: View Statistic page** 25](#_Toc15388192)

# **Chapter I | Introduction**

## **1.1 Purpose**

The purpose of the software design document (SDD) is to design the detailed structure of the system according to the software requirement specification (SRS). It intended to help stakeholders in the project understands the detailed design of the system using class diagrams, sequence diagrams, and user interface designs.

## **1.2 Intended Audience and Reading Suggestions**

### **1.2.1 Intended Audience**

This software requirement specification will benefit to audiences as followed:

**Development Team**

* The software requirement specification will help everyone in developer team has the same understanding or overall scope of the project.
* The software requirement specification will help the developer to verify and validate the software product that matches with the requirements in this document or not.
* The software requirement specification will help developers to develop project more efficiency.
* The software requirement specification will be required when some changing occurs. Developer will know where get effect and know where to change.

**User**

* The software requirement specification will help the user to know the process of the software product.
* User can check that the requirements in software requirement specification match with their requirement or not.
* The software requirement specification can help the user to discuss with developers easier.

## **1.3 Project Scope**

Elephant Learn is a web application that uses web service to fulfill its functionality. It let admin view the percentage of the topic on sinthorn section in Pantip. If the comments are negative admin can delete the comments. Admin can view the statistic of information, which is the number of users, date, and time of the comment. Moreover, admin can view all data which is in the system and data that have deleted before.

## **1.4 Acronyms and Definition**

### **1.4.1 Acronyms**

SRS Software Requirement Specification

SDD Software Design Document

CD Class Diagram

SD Sequence Diagram

UI User Interface

CY Mr. Chamnol Yin,

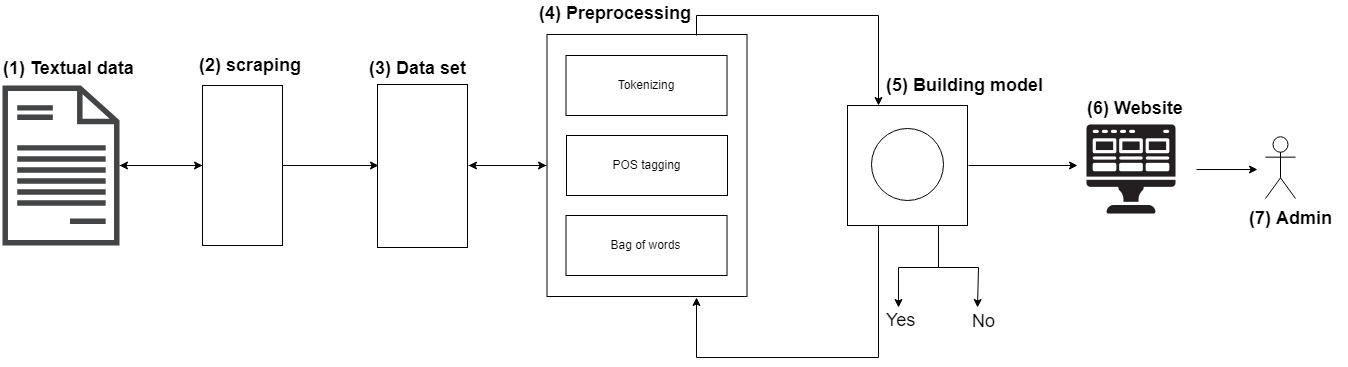
PM Miss Pakarat Matmarurat,

PT Dr. Pree Thiengburanathum

### **1.4.2 Definition**

|  |  |
| --- | --- |
| **Name** | **Definition** |
| **Feature** | Transformation of input parameters to output parameters based on a specified algorithm. It describes the functionality of a product in the language of the product. Used for requirements analysis, design, coding, testing or maintenance. |
| **Sequence diagram** | A sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. |
| **Class diagram** | Class diagrams are an aspect of UML that describe a static design of the objects, and their relationships to each other, in an application. During analysis, class diagrams may just be the names of objects and how they interact, but as the design develops the details of each class, including attributes and methods, are displayed in the diagram. A class is represented by a rectangle divided into three rows. The top row contains the name of the class. The middle row lists the attribute names, while the third row lists the method names for the class. |
| **User interface** | User interface (UI) is everything designed into an information device with which a human being may interact -- including display screen, keyboard, mouse, light pen, the appearance of a desktop, illuminated characters, help messages, and how an application program or a Web site invites interaction and responds to it. |

# **Chapter II | System Architecture**



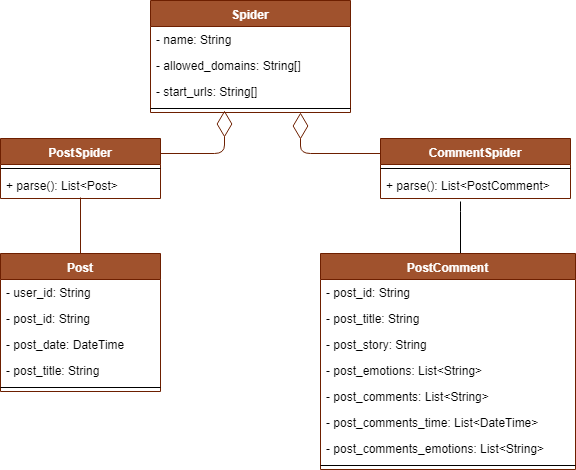
**Figure 1**: System Architecture

According to figure 12, system has been illustrated as follow: (1) we retrieve the raw data from sinthorn section on Pantip blog and (2) scraping the data. Once get the (3)data set then put it to (4) preprocessing in this part it’s will going to convert the data by basic tokenizing which typing the word and remove outliner which is white space. POS tagging to extracting relations between words. After using the bag of words, the data will represent as a number in matrix form then data analysis and got (5) model. When we finish the back end then connected with (6) front end.

# **Chapter III | Detailed Design**

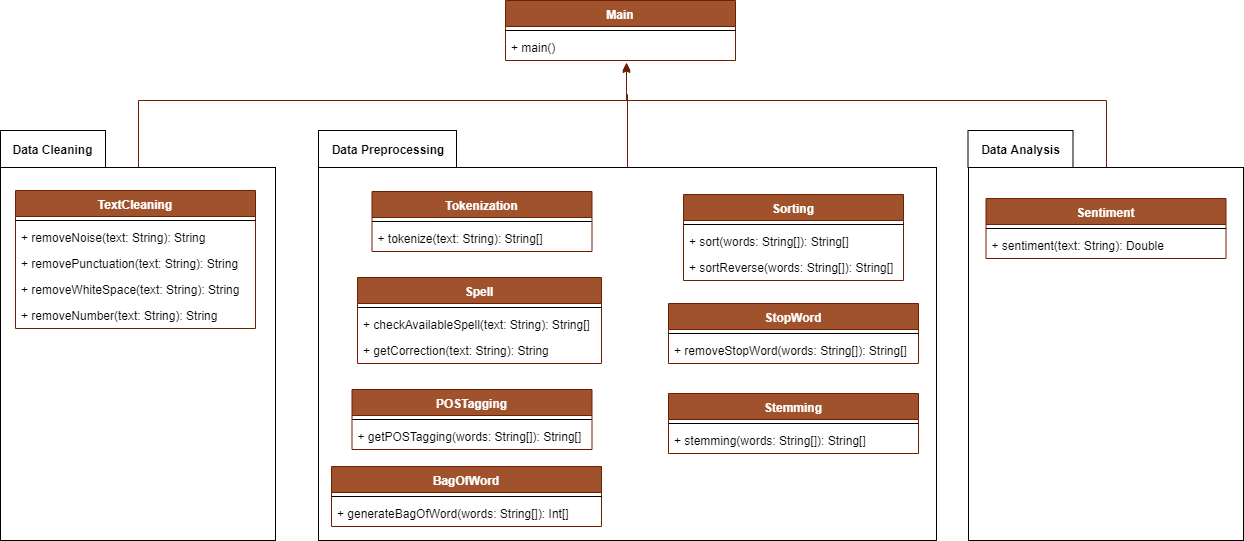
## **3.1 Class Diagram**

**Feature#01:**

****

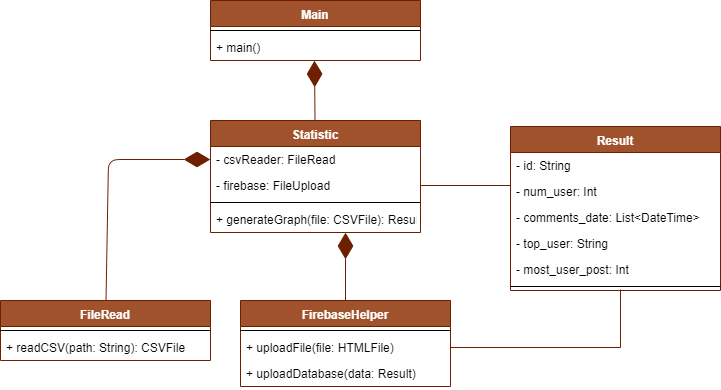
**Figure 2**:Class Diagram Feature#01

**Feature#02:**

****

**Figure 3**:Class Diagram Feature#02

**Feature#04:**

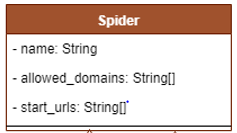
****

**Figure 4**:Class Diagram Feature#04

## **3.2 Class Diagram Description**

* **Feature#01**

### **3.2.1 Class name: CD-01: Spider**

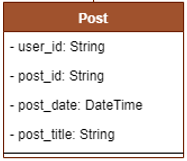


**Figure 5**: Spider

**Attributes:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | name | Name of spider class | String |
| 2 | Allow\_domains | The array website domains that spider will be able to do scraping | String[] |
| 3 | Start\_urls | The array of websites’ URLs that spider will do scraping | String[] |

### **3.2.2 Class name: CD-02: Post**

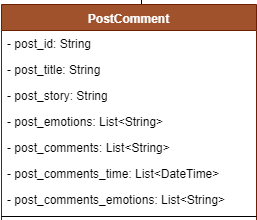


**Figure 6**: Post

**Attributes:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | user\_id | The unique key to identify post’s user of Pantip | String |
| 2 | post\_id | The unique key to identify post | String |
| 3 | post\_date | Time that post has been posted | DateTime |
| 4 | post\_title | The title of post | String |

### **3.2.3 Class name: CD-03: PostComment**



**Figure 7**: PostComment

**Attributes:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | post\_id | The unique key to identify post | String |
| 2 | post\_title | The title of post | String |
| 3 | post\_story | The post’s detail/description | String |
| 4 | post\_ emotion | The number and type of emotions that has been reacted of post | List<String> |
| 5 | post\_comments | The comments of post | List<String> |
| 6 | post\_comments\_time | The comments of post time | List<DateTime> |
| 7 | post\_comments\_emotions | The number and type emotions that has been reacted of comments | List<String> |

### **3.2.4 Class name: CD-04: PostSpider**



**Figure 8**: PostSpider

**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | parse() | Method to scrap the post from pantip website | List<Post> |
| Return: the list of scraped post |

### **3.2.5 Class name: CD-05: CommentSpider**



**Figure 9**: CommentSpider

**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | parse() | Method to scrap the comment from pantip website | List<PostComment> |
| Return: the list each post’s comments |

* **Feature#02**
* **Data Cleaning**

### **3.2.6 Class name: CD-06: Main**

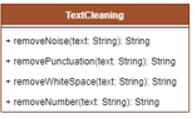


**Figure 10**: Main

**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | main() | Main method of system |  |
| Return: - |

### **3.2.7 Class name: CD-07: TextCleaning**



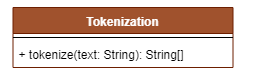
**Figure 11**: TextCleaning

**Methods:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | removeNoise(text: String) | This method create to remove the noise of the textual comment. This method will remove such as html tag from textual comment | String |
| Return: text without noise |
| 2 | removePunctuation(text: String) | This method create to remove punctuation of comments such as question mark .etc. | String |
| Return: text without punctuation |
| 3 | removeWhiteSpace(text: String) | This method create to remove the white space of textual comments. | String |
| Return: text without white space |
| 4 | removeNumber(text: String) | This method create to remove the number on comments. \*\*optional | String |
| Return: text without number |

* **Data Preprocessing**

### **3.2.8 Class name: CD-08: Tokenization**

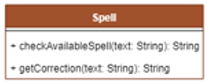


**Figure 12**: Tokenization

**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | Tokenize(text: String) | Method to split sentence/paragraph of text to be separated words. | String |
| Return: Array of tokenized words |

### **3.2.9 Class name: CD-09: Spell**



**Figure 13**: Spell

**Methods:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | checkAvailableSpell(text: String) | This method create to check the available correction spell of words of text | String[] |
| Return: the array of available/probable words |
| 2 | getCorrection(text: String) | This method create to get the correction of words | String |
| Return: the most probable correction of words/sentence of text |

### **3.2.10 Class name: CD-10: POSTagging**



**Figure 14**: POSTagging

**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | getPOSTagging(words: String[]) | This method create to get part of speech tagging of array of words | String[] |
| Return: the array of path of speech of words |

### **3.2.11 Class name: CD-11: BagOfWord**



**Figure 15**: BagOfWord

**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | generateBagOfWord(words: String[]) | This method create to generate Bag-of-words which form in array of integer that represent the present of word from system’s corpus in the input words. | Int[] |
| Return: the array of bag of word in form of array of integer. |

### **3.2.12 Class name: CD-12: Sorting**



**Figure 16**: Sorting

**Methods:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | sort(words: String[]) | The method to sort the array of word to be in order. | String[] |
| Return: the ordered words array |
| 2 | sortReverse(words: String[]) | The method to sort the reverse order of words in the array | String[] |
| Return: the reverse ordered array |

### **3.2.13 Class name: CD-13: StopWord**



**Figure 17**: StopWord

**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | removeStopWord(words: String[]) | This method create to remove stop word to reduce the repetitive of similar words | String[] |
| Return: array of word with less repetitive |

### **3.2.14 Class name: CD-14: Stemming**



**Figure 18**: Stemming

**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | stemming(words: String[]) | This method create to stemming the words which reduce the similarity of words | String[] |
| Return: array of word with less similarly |

* **Data Analysis**

### **3.2.15 Class name: CD-15: Sentiment**



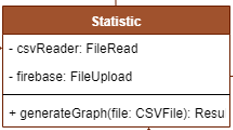
**Figure 19**: Sentiment

**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | Sentiment(text: String) | Method to calcutalte the percentage of sentimental analysis of text | double |
| Return: the percentage of sentimental analysis of text |

* **Feature#04**

### **3.2.16 Class name: CD-16: Statistic**



**Figure 20**: Statistic

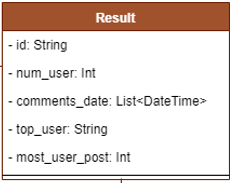
**Attributes:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | csvReader | Object to read csv file | FileRead |
| 2 | firebase | Object to upload file to database (firebase) | FileUpload |

**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | generateGraph(file: CSVFile) | This method create to generate graph of the statistic information | Result |
| Return: the result of statistic calculation |

### **3.2.17 Class name: CD-17: Result**

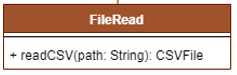


**Figure 21**: Result

**Attributes:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | id | The unique key to identify result | String |
| 2 | num\_user | Total number of users who has posted | int |
| 3 | comments\_date | Time of comments have been posted | List<DateTime> |
| 4 | top\_user | User with most posts/comments number | String |
| 5 | most\_user\_post | The total number that top user has posted | int |

### **3.2.18 Class name: CD-18: FileRead**



**Figure 22**: FileRead

**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | readCSV(path: String) | This method create to read CSV file from input path of csv file | CSVFile |
| Return: the data from csv file |

### **3.2.19 Class name: CD-19: FirebaseHelper**



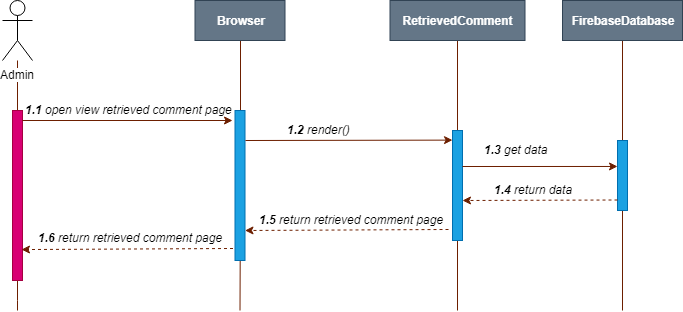
**Figure 23**: FirebaseHelper

**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | uploadFile(file: HTMLFile) | This method create to upload HTML file to database (firebase) | HTMLFile |
| Return: - |
| 2 | uploadDatabase(data: Result) | This method create to upload the result to the database in format of JSON object |  |
| Return: - |

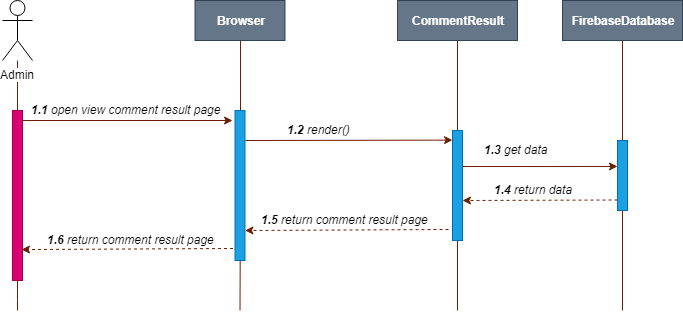
## **3.3 Sequence Diagram**

### **3.3.1 SD-01: Admin can view the retrieved comments from Pantip blog.**

****

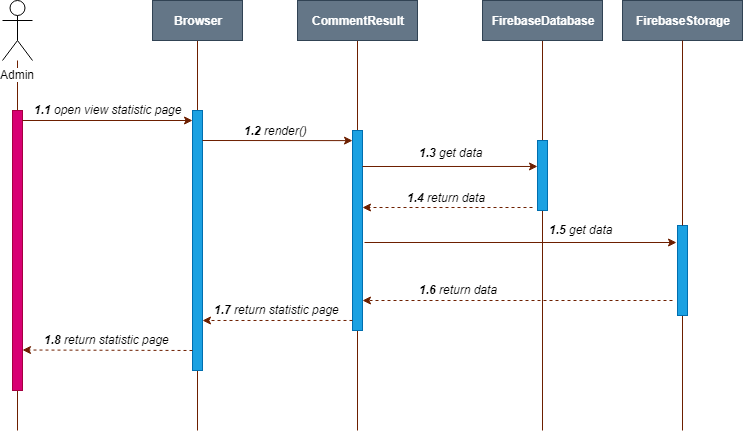
**Figure 24**: SD-01

### **3.3.2 SD-02: Admin can view the result of sentimental analysis of each comment in the topic.**

****

**Figure 25**: SD-02

### **3.3.3 SD-03: Admin can view the statistic of information, which is the number of users, date, and time of the comments.**

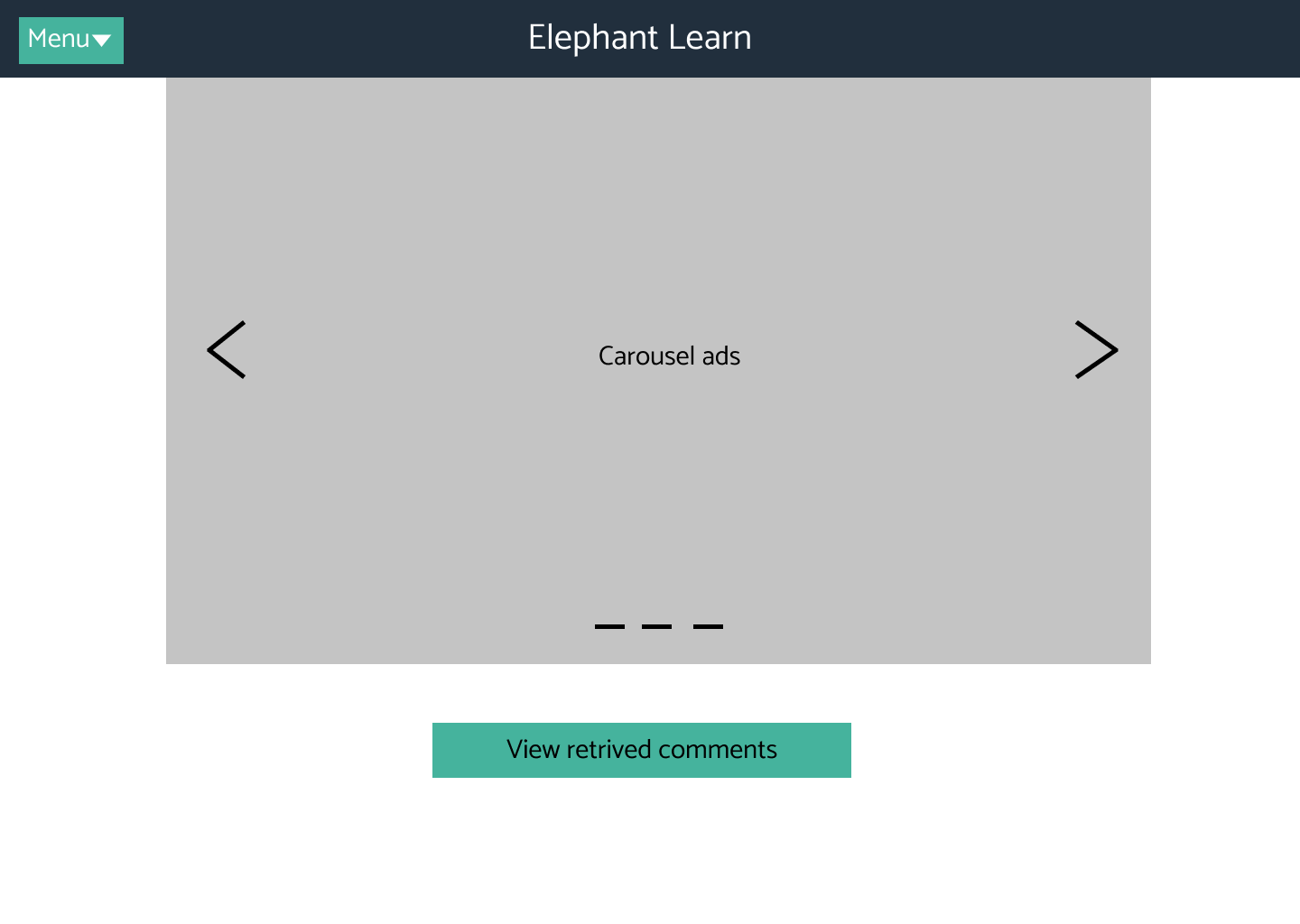
****

**Figure 26**: SD-03

# **Chapter IV | User Interface Design**

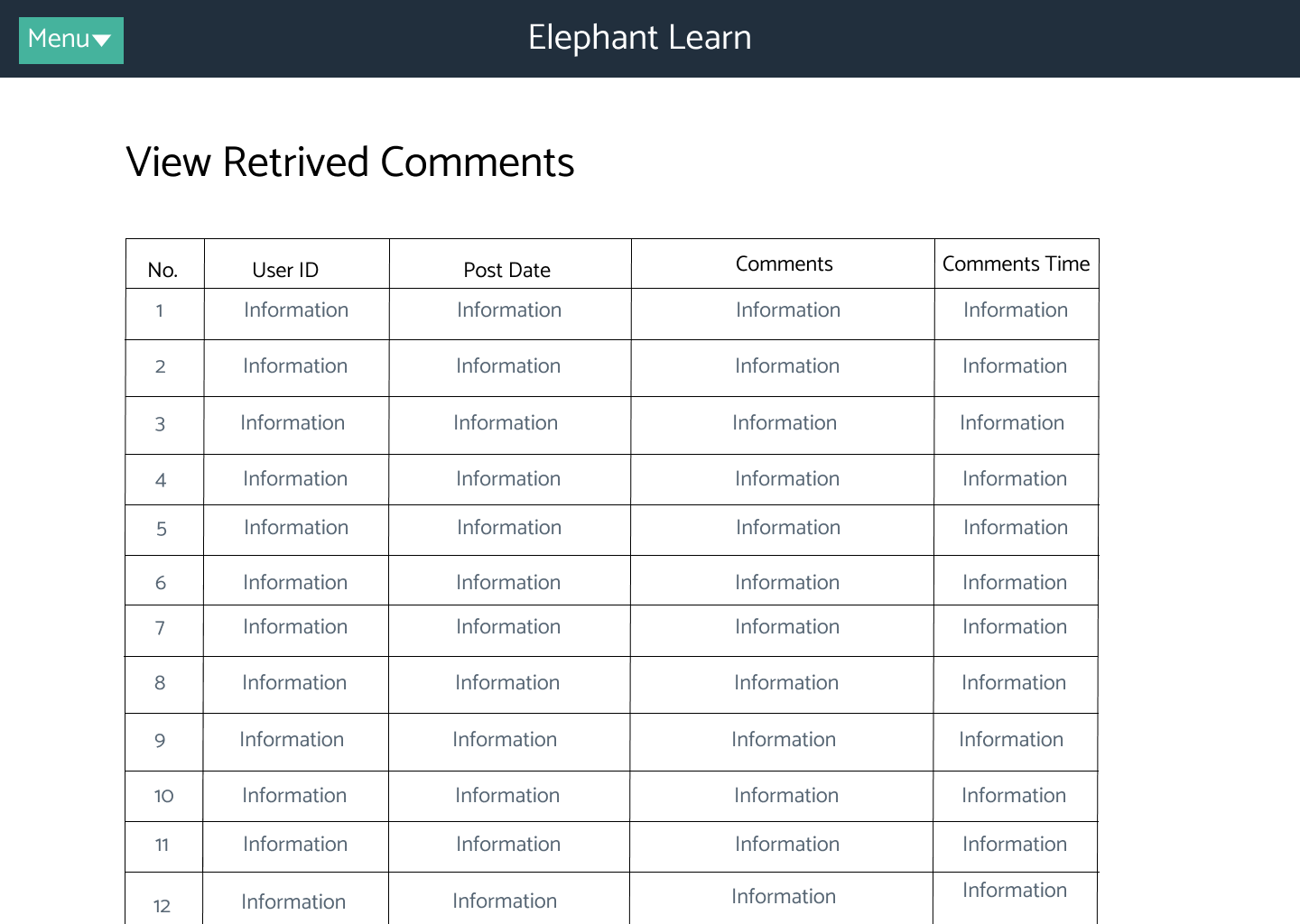
## **4.1 User Interface (UI)**

### **4.1.1 UI-01: Home page**



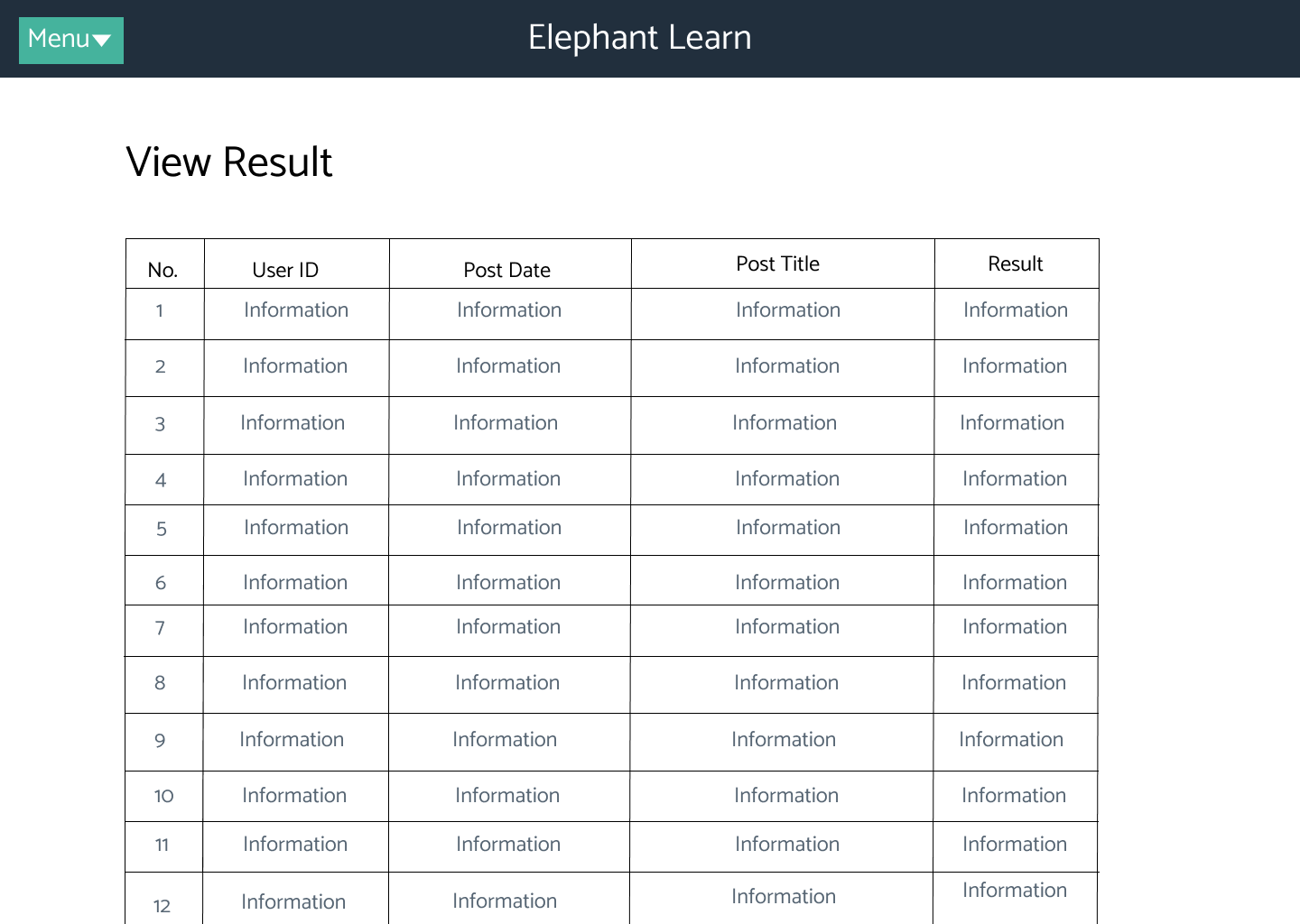
**Figure 27**: UI-01

### **4.1.2 UI-02: View Retrieved Comments page**



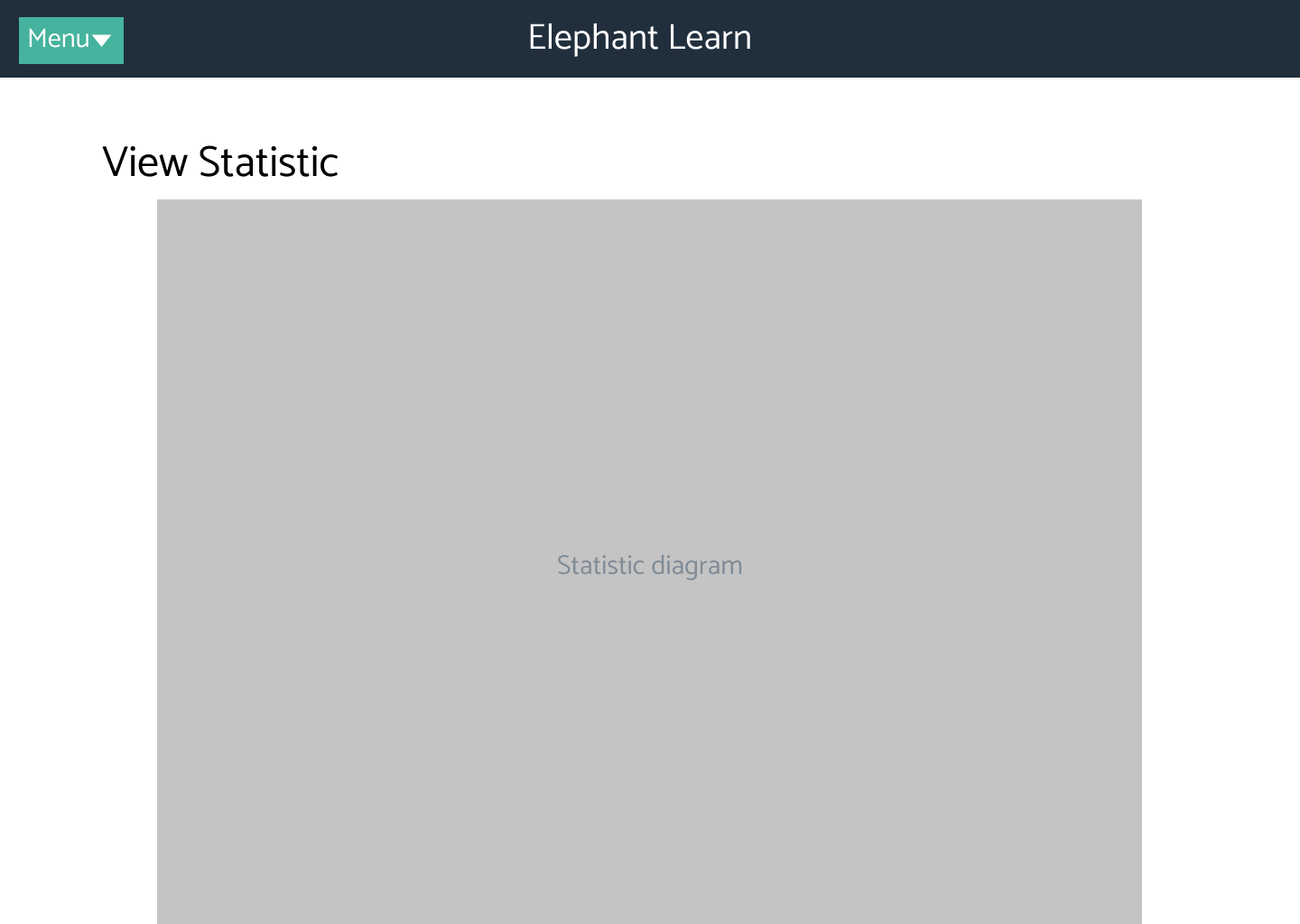
**Figure 28**: UI-02

### **4.1.3 UI-03: View Result page**



**Figure 29**: UI-03

### **4.1.4 UI-04: View Statistic page**



**Figure 30**: UI-04