BUGCOLI WEB APPLICATION

Theerawee Rangram
Piyapat Russamitinakornkul
Sirawat Khiewsakul

BACHELOR OF ENGINEERING PROGRAM IN SOFTWARE ENGINEERING
INTERNATIONAL COLLEGE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2015

Thesis – Academic Year 2015

B.Eng. in Software Engineering

International College, King Mongkut's Institute of Technology Ladkrabang

Title: Bugcoli Website

Authors:

1. Mr.Theerawee Rangram Student ID 55090026
2. Mr.Piyapat Russamitinakornkul Student ID 55090031
3. Mr.Sirawat Khiewsakul Student ID 55090048

Approved for submission

.....

(Dr. Isara Anantavrasilp)

Advisor

Date:

02/08/2016

Abstract

Nowadays, most of Thai programmers only search for answers when they face some programming bugs or errors. However, it is unlikely for them to create or answer issues. This is due to the language barriers, which most of Thai programmers do not feel comfortable to discuss or explain issue in English. Therefore, this project aims to build Thai community where Thai programmers come in and share knowledge to each other. A part from create question and answer, this project also provides some great features as following. It allows user to reply on answer, bookmark questions or keep up-to-date on that questions, vote for question or answer to support his knowledge, Thai and English search, pick best answer for certain question. In addition, this website is be able to list most likely questions that user would interest and encourage user to visit. Furthermore, this website can evaluate top three strongest topic of user which show on each user profile page. Next, notification is also provided in this website, user will be notified for updated information every time someone vote on his question or answer, answer his question, reply his answer, and select his answer as best answer. Moreover, this web application is compatible with any screen size of display.

Acknowledgement

To begin with, we desire to express our sincere thanks to our project advisor, Dr. Isara Anantavrasilp for spending so much time taking the trouble to help us, and considerable suggestion. We are grateful for his advice, encouragement, and invaluable experiences the he shared to us, not only the project consultation but also many morals and recommendation in real life. Thanks for his guidance how to solve conflicts we confronted, guideline to pursue our dream regardless obstacles we face, lastly guideline in advancing our presentations, thesis and Bugcoli website.

Last but not least, we delightedly acknowledge our parent for fully supporting us to finish this thesis and all professors from the faculty of International College, KMITL who encourage us to do projects every semester which improve our capability to be super strong to work on every environment, any pressure. Without their support and recommendation, we would not be able to finish the thesis and have been succeeded so far.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Today, anyone who owns a computer can try to create his own programs with only a small knowledge of coding. However, no matter how easy it gets, errors and bugs always occur. Sometimes he may not be able to resolve the issue on his own. Also, the programmers may simply not know how to solve a problem or achieve a desired task.

Fortunately, with the Internet, it is easier to solve such problems. There are many popular websites on the Internet that allow users to post questions, write answers to the questions, and reply those answers. However, these websites use English as a main language for discussions. Therefore, Bugcoli aims to help Thai programmers both at amateur or professional level to be able to discussions problems encountered during programming in Thai. Bugcoli will eliminate language barriers and help programmers to expand their discussion topics vast and various.

On Internet networks, there are no proper discussion communities for Thai programmers yet. They have to rely on international community websites and struggle with fluent English. Therefore, Bugcoli is a website dedicated for Thai programmers. A Thai community for discussing programming issues. People with Thai language skills can participate in help each other to make programming easier in Thai language.

1.2 Objectives

Bugcoli aims to become a website which allows its member to ask questions and reply answers regarding programming issues in Thai language. When interesting quesions are present, members can vote by giving scores to support and agree that the discussion is useful to others. Our website aims to connect Thai programmers together to help each other and improve their community.

1.3 Scope of Work

The procedure of implementing and developing Bugcoli can be categorized into three phases:

1. Study Phase: Study similar operating websites, which is an international community and is using English language to define a border of discussion topics for Bugcoli. Then, study many various website development tools, to pick the best tool which suits the demands of Bugcoli's funcitonality.

2. Design Phase: Establish a plan of the project development and design website content specifications such as database diagram and webpage mockup.

3. Implementation Phase: Develop the website using all results from the first two phase.

1.4 Contributions

Bugcoli will feature its own unique search engine. Users will be able to search for questions in Thai. Despite the fact that there exists many cross-language search engines, Bugcoli's will be an accurate Thai search engine. It will focus mainly searching only in Thai language.

We will build a community where Thai programmers can gather and discuss or exchange programming issues.

Chapter 2

Literature Review

This chapter reviews various web blogs with different reputations. 2.1, existing related works in detail which explain about features are explained. Section Section 2.2 describes webblog fundamental features. In section 2.3 is a comparison table of features between Bugcoli and other existing web blogs.

2.1 Related works:

2.1.1 what is webblog?

Webblog is a kind of website that allow user to write user story, knowledge, or to ask something from other user. The point of WebBlog can be a memory for user himself, or sharing to other users.

2.1.2 what is blog?

Blog is one of user-created-topics in Webblog. Mostly, it contain title of topic, content of topic, and user information of whom create. Nowadays people use Blog as a tool to publicize news, inform customer. For example, one of the reasons why Webblog created in e-commerce Webblog is that e-commerce has a high competitive rate. Therefore, in order to convince customer, Blog is made to explain about product and build trust by the comment of people who has bought the product. These are popular Webblogs showing below.

Stack Overflow Website

Stack Overflow is a programmer community, where people ask about programming problem, share knowledge and help each other fixxing bugs in only English language.

- For the Server-Side, it uses Microsoft ASP.NET, which is an open-source server-side web application framework written in C#.
- For the Client-Side, it uses jQuery, which is a JavaScript framework. Its purpose is to simplify the JavaScript on its website. Lastly, Stack Overflow uses SQL Server 2008 as its information storage [2]

From figure 2.1, it is a StackOverflow home page. A question in the homepage of StackOverflow shows its title, the amount of votes, answers and views and tags that are related

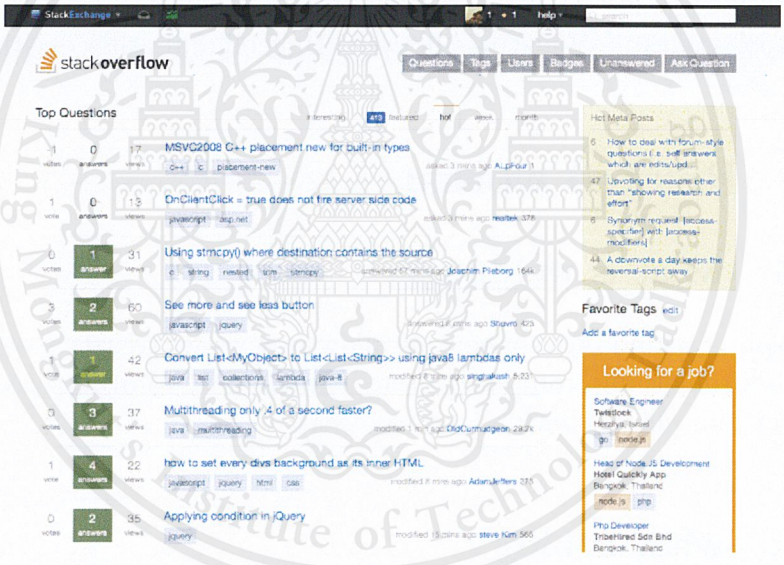


Figure 2.1: StackOverflow home page

From figure 2.2, Creating a blog in StackOverflow require title, description and tags.

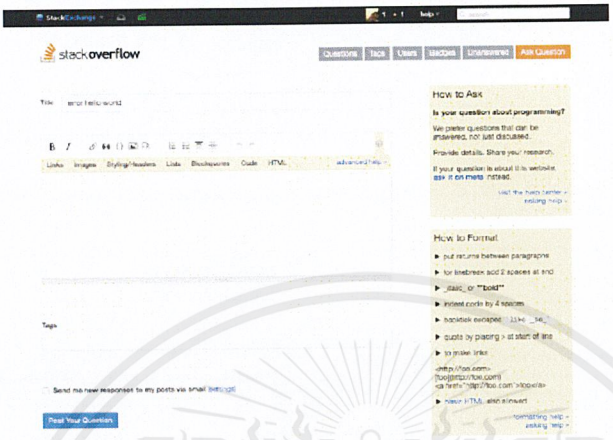


Figure 2.2: StackOverflow Creating Blogs page

From figure 2.3, it is a StackOverflow blog page. User can give feed back for a blog by upvoting or devoting at the top left of the page. The number in ther middle show the total of blog vote.

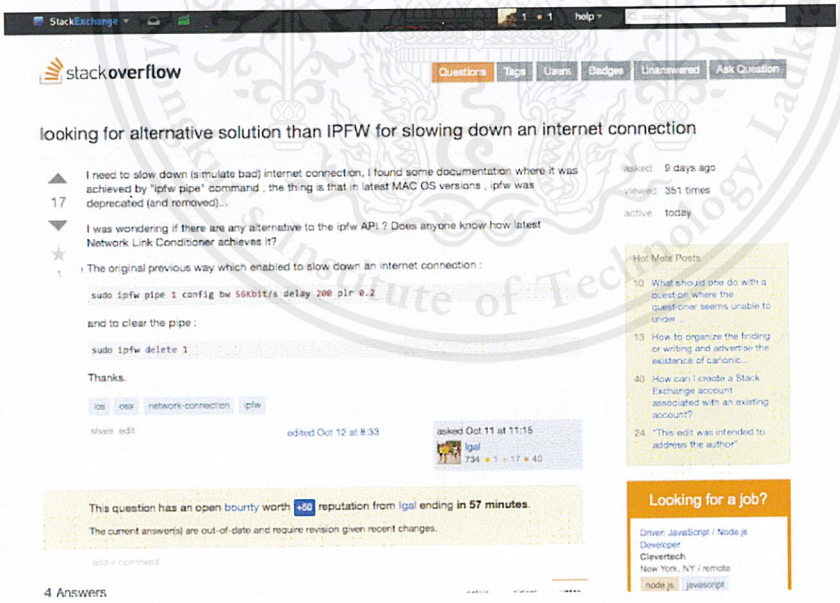


Figure 2.3: StackOverflow blog detail

From figure 2.3, The correct signal at the top left of the page is meaning that the comment is selected as the best answer for the figure question

Figure 2.4: StackOverflow blog detail

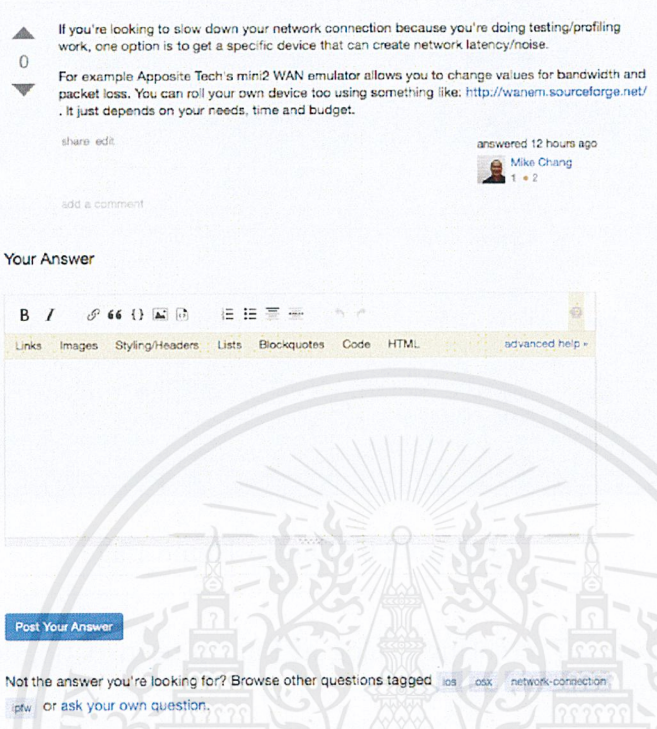


Figure 2.5: StackOverflow comments

Pantip Website

Pantip is currently one of the most popular online webblog for Thai community network. The website provides a space for Thai people to ask questions, share ideas, and discuss with others in various topics. For the technologies being used, PHP is used as Server-side. For database, Pantip.com used to operate wih MySQL database. Recently, they have changed it to MongoDB instead [9, 4].

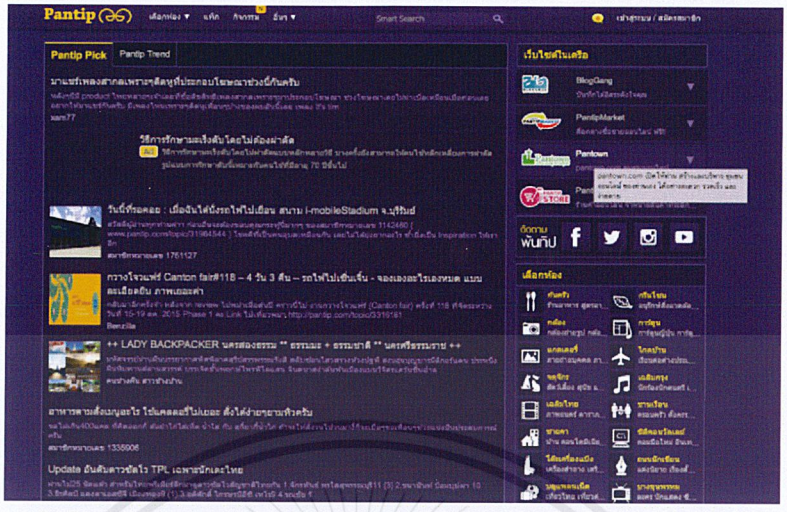


Figure 2.6: Pantip Home Page

From figure 2.7, creating a blog in Pantip needs title, description and at most five tags from categories

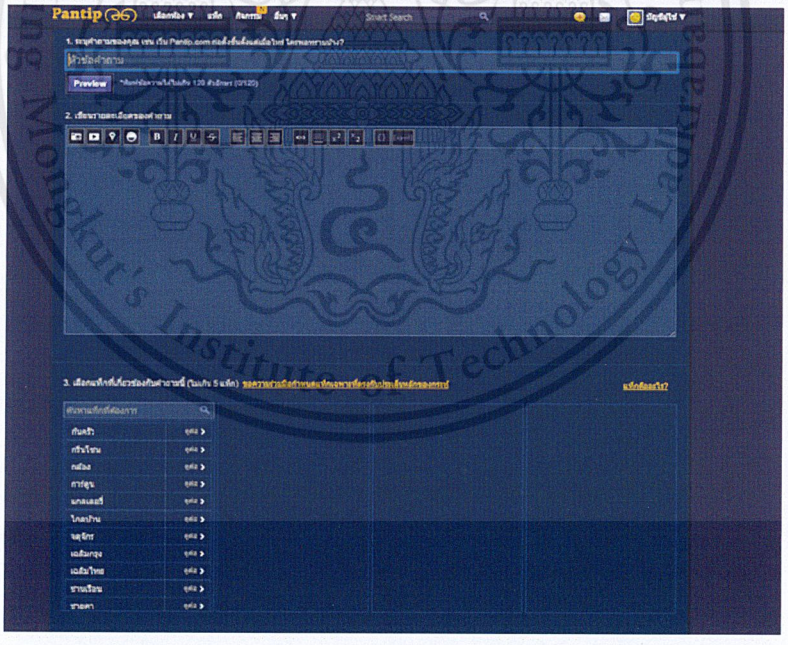


Figure 2.7: Pantip Creating Blogs page

From figure 2.8, it is Pantip blogs page. Tags, votes and emotions are considerated in a post.

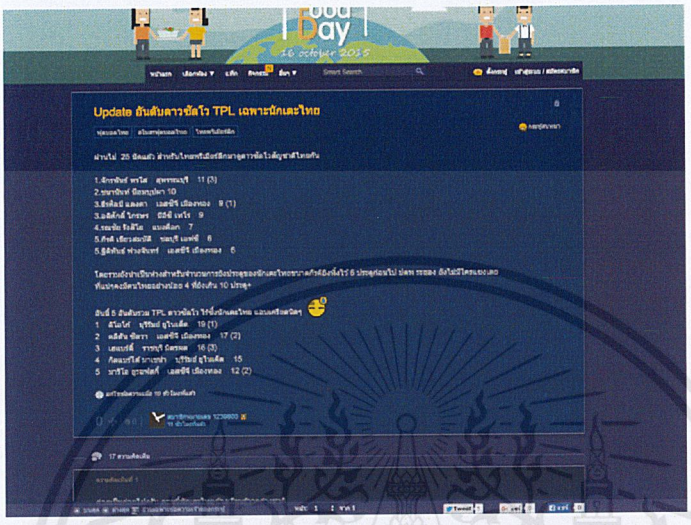


Figure 2.8: Pantip blogs detail page

From figure 2.9, Replying to a comment in Pantip

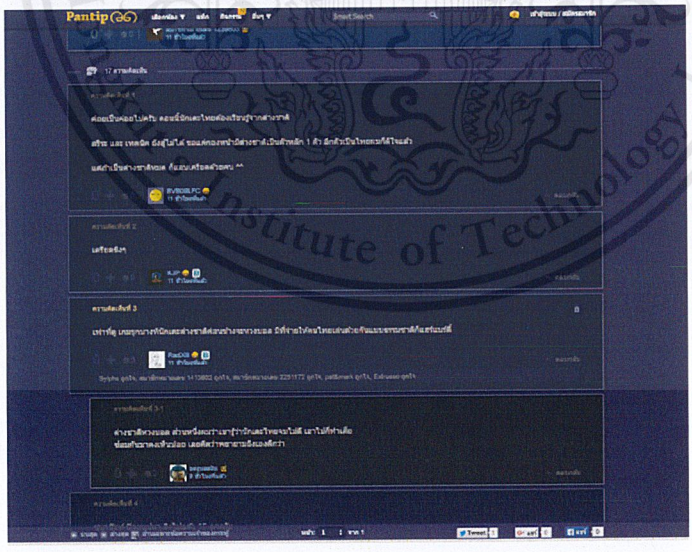


Figure 2.9: Pantip Blog detail

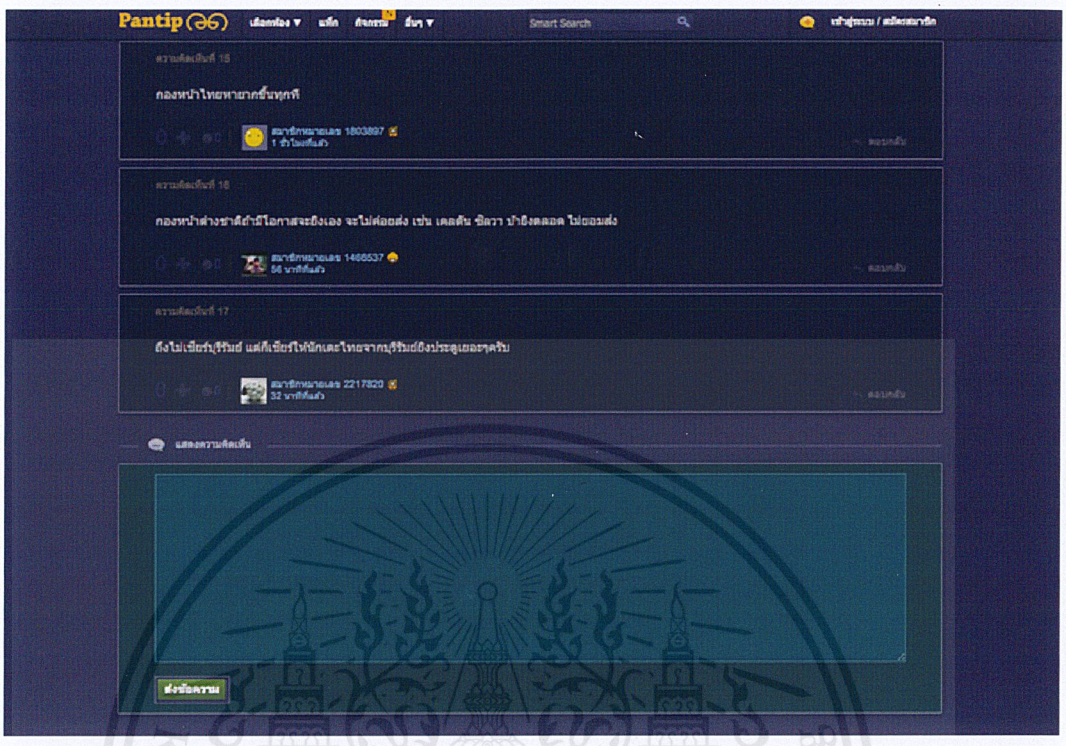


Figure 2.10: Pantip Comments

Blognone Website

Blognone is a Thai webblog which is focusing on sharing technology and science information from all around the world. The information is summarized easily for readers to understand. This website allows its members to post information as well as make comments. There are also interviews from experienced or successful technologist, programmers, etc. For web technologies of this website, it is restricted from revealing them.

From figure 2.11, it is Blongnone homepage Share feature at the top of each blog is available for a post in Blognone homepage

Figure 2.11: Blognone home page

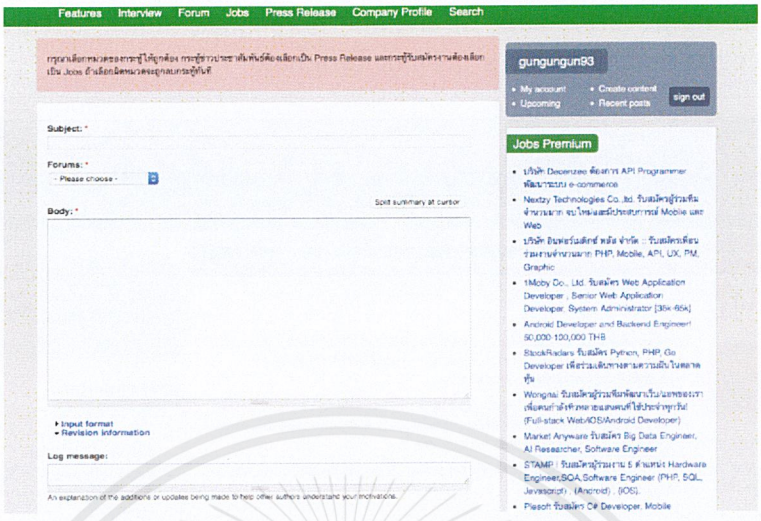


Figure 2.12: creating blogs in Blognone:

From figure 2.13, This page shows when a user click to view a blog

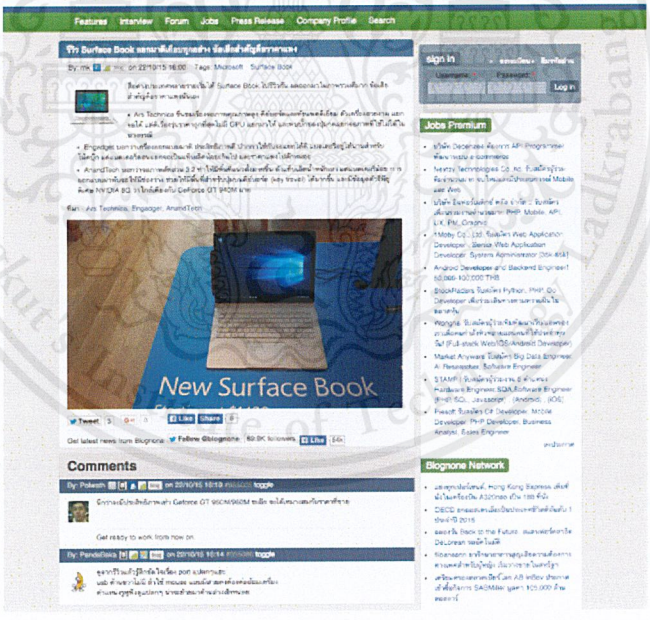


Figure 2.13: blognone Blog Detail

2.2 Integrated Bugcoli Features

1. Sign-up and sign-in: Webblog application can be registered user-account and login as member user to be able to access more feature in application.

2. Login with social network: An application has additional permitted accounts from social networkds such as Facebook, Google Plus, Twitter, Github, or Linkedin.

3. Create questions: Users can create question to ask for help regarding programming issues

4. Create answers: Users can answer questions.

5. Reply on answers: Users can reply answer.

6. Upvote or devote questions or answers: Users can upvote if the question or comment is educational, and devote them if they are unrelated to programming topics. Questions' or Answer' owner will obtain 5 points from other users' votes.

7. Select the best answer: Owner of question can pick the best answer of his questions to push up the answer on top for making answer outstanding over non-best answer. In addition, the select answer's owner will obtain 5 point for sharing a great knowledge.

8. User ranking: Users are classified into different levels calculated from user's points. Points are obtained from votes given by other users, answers or get selected as the best answer.

9. Thai language search question, and answer: Users can search for related information in questions in Thai language

10. English language search question, and answer: Users can search questions in English language

11. Notifications: Notifications notify users on updates to user's question or answer. For example, when a user give a answer to a question. The question owner will get information of notification that it is a answer notification with giving user information such as user's picture, name.

12. User profile management: Profile details include information of contacts such as name, and display image. This feature allows users to change and update their information.

13. Share questions to social network: Questions can be shared to social networks namely Facebook, Google Plus, Twitter

14. Bookmark questions: Users can keep track of their favorite questions with bookmark feature.

15. Sort questions: Questions can be sorted according to number of bookmark, date, tags, most views, total votes, and total answers.

16. Show top three proficient topics of user: In user profile, it shows the best three tags that user is good at.

17. Show collection of question that user has created, answer, vote, and bookmark: Whever user create a question, answer a question, vote a question or answer, bookmark a question, they all save in user profile.

18. Calculate number of views in each question Everytime user visite a question detail, it will increase number of view in that question if user visite the question at the first time

19. Suggest questions: The program provide suggest questions for user to go answer them. The suggest questions are gathered from viewing question detail.

2.3 Comparison

	Bugcoli	Pantip	StackOverflow	Blognone
Sign-up and sign-in	✓	✓	✓	✓
Login with social network	✓	✓	✓	
Create questions	✓	✓	✓	✓
Create answers	✓	✓	✓	✓
Reply on answers	✓	✓	✓	✓
Upvote or devote questions or answers	✓		✓	
Select the best answer	✓	✓	✓	
User ranking	✓		✓	
Thai language search blog, and answer	✓	✓		✓
English language search blog and answer	✓	✓	✓	✓
Notification	✓		✓	
User profile management	✓	✓	✓	✓
Share blogs to social network	✓	✓	✓	✓
Bookmark questions	✓	✓	✓	
Sort questions	✓		✓	
Show top three proficient topics of user	✓			
Show collection of question that user has created, answer, vote, and Bookmark	✓	✓	✓	
Calculate number of views in each question	✓		✓	
Suggest questions	✓			

Table 2.1: Site Features Comparison

Chapter 3

Background Knowledge

This chapter describes background knowledge of implementing the Bugcoli project. There are three topics to be covered. The first topic is the system design, it describes information about different types of diagrams and their specifications of this project. The second topic provides details of chosen development tools and its decision factors.. Finally, the background knowledge for implementing search algorithm for search feature in Bugcoli.

3.1 Unified Modeling Language Diagrams

A Unified Modeling Language (UML) is a standard notation systems for modeling of real-world objects as a plaining step in developing an object-oriented design methodology. It provides various model elements to visualize various of system's architecture.

3.1.1 Use case Diagram

In UML, use case diagram is used to show all user's interactions and their relationships with subsystem. In a large system, a user is defined as an actor while a subsystem is defined as a use case. The use case diagram for Bugcoli can be thought of how users will perform tasks on the website. It outlines, from a user's point of view, a system's behaviour as it responds to a request beginning with a user's goal and ending when that goal is fulfilled.

3.1.2 Package Diagram

In the UML, a package diagram represents structure of the overview designed system and describes its dependencies between each other. In Bugcoli, the package diagram describe relasionship between server side, client side, database, and search in big picture.

3.1.3 Class Diagram

A class diagram, in UML, is a static structure diagram used to illustrate the system's classes and the relationships between them. Each class presents information including its name, attributes, and operations. In Bugcoli, the class digram generally describes the compoennt of the server side, client side, database, search components, and how are they related to each other.

3.1.4 Sequence Diagram

In UML, the sequence diagram is a diagram that represent an interaction between classes. The interactions of classes identify method calling and result getting. In Bugcoli, sequence diagram show events/messages between modules in system occur in time

3.2 Development Tools and Software Libraries

3.2.1 Go

Go is an open source programming language developed at Google and designed to help build simple reliable software systems. We use Go to create web servers to handle all request from client side and various additional functionalities.

3.2.2 React/Flux

React (React.js or ReactJS) is an open source Javascript library providing a view for data rendered as HTML. React views are typically rendered using components that contain additional components specified as custom HTML tags.

Flux provide a model and control for data render as HTML. It is the application architecture that is used for building client-side web applications. It complements React's compassable view components.

- **Webpack**

  Webpack is a module bundler that solves a lot of manual work in Gulp to achieve with same results. It supports advanced functionality such as hot module reloading (instant updates of ReactJS components without refresh), lazy loading (load bundles as you need), bundle splitting (separate app/vendor bundles), hashing (to bust cache) and source maps (so it's easy to debug minified versions).

3.2.3 HTML/CSS

HTML and CSS are two of the core technologies for building Web pages. HTML provides the structure of the page. CSS provide graphics and scripting, HTML and CSS are the basis of building Web pages and Web Applications.

3.2.4 Sass

Sass is dynamic style sheet language that interpreted into CSS. They provides the following mechanisms ,namely, variables, nesting, mixins, operators and functions to create CSS.

- **Gulp**

  Gulp is a task/build runner for development. It allows to do a lot of stuff within development workflow. it can compile Sass/Less files, uglify and compress js files and much more.

3.2.5 Sublime text

Sublime text is a cross-platform source code editor with a Python application programming interface (API). It natively supports many programming languages and markup languages. We use Sublime text for implementing and editing source code.

3.2.6 MongoDB

MongoDB is a cross-platform document oriented database. MongoDB does not use the traditional table-based relational database structure, but in stead, it favors of JSON-like documents with dynamic schemas called "Collections", making the integration of data in types of application of data in certain types of application easier and faster.

3.2.7 Robomongo

Robomongo is a shell-centric cross-platform open source MongoDB management tool. It provides an interface for database viewing and modifying.

3.2.8 TeXstudio

TeXstudio is a cross-platform open source LaTex editor. TeXstudio is a LaTeX IDE that provides modern writing support, such as interactive spelling checker, code folding, and syntax highlighting.

3.2.9 FileZilla

FileZilla is a cross-platform FTP(file transfer protocol) application. We use this tool to transfer our projesct file to server.

3.3 Search Feature

In Bugcoli, searching is a feature which allows all users to find questions on the website by entering some keywords ranging from one word to a short text. The text is called a "User Query". Generally, search engines find webpages with contents corresponding to the query [1]. It also sorts all retrieved questions according to relevancy. There are three steps for implementing Bugcoli Search:

1. Indexing
2. Query (Retrieving)
3. Ranking Results [7]

3.3.1 Implementation

Step 1: Indexing

Bugcoli Search retrieves a question within its own database. Therefore, it does not require a crawler to explore the Internet like general search engines [7]. In contrary, whenever a new question is posted to the website, the server shall store the contents of the question.

Inverted Index is a storage containing a list of words. Each word is mapped to the identification (ID) of each question containing the word. However, questions are stored as a list of words. Therefore, the index is considered inverted and called an inverted index. An inverted index is required to store for each word at list of questions containing the word [1]. For example of storing an inverted index, consider if the website contains three questions with the following content:

1. The only way not to think about money is to have a great deal of it.
2. When I was young I thought that money was the most important thing in life; now that I am ld I know that it is.
3. A man is usually more careful of his money than he is of his principles.

The inverted index shall contain information as follows.

Words	Document
a	1, 3
About	1
am	2
Careful	3
deal	1
great	1
have	1
he	3
his	3
i	2
important	2
in	2
is	1, 2, 3
it	1, 2
know	2
life	2
man	3
money	1, 2, 3
more	3
most	2
not	1
now	2
of	1, 3
old	2
only	1
principles	3
than	3
that	2
the	1, 2
thing	2
think	1
thought	2
to	1
usually	3
was	2
way	1
when	2

Table 3.1: Inverted Index Example [1]

Bugcoli's Database contains a collection called "Inverted Index" to store documents of each word mapped to the question ID where its content contains the word. Each word may be found in multiple positions in a question. However, only one occurrence is stored, and the rest are ignored. In addition, each word in language with alphabet cases is also converted to small case to make the word non-case-sensitive.

Step 2: Query Index

Query is a very short text ranging from one to multiple words. This text is used as a key to search for the required information. Retrieved information in this step will be called "Search Results". Questions are matched . There can be different types of query. One example is a phrase query, where users enter a text between double quotes. This kind of query requires that exact phrase within the double quotes also appears in the content of a question [1].

However, in Bugcoli, the only query type is normal, meaning that there are no special conditions. Bugcoli's search function only matches contents of questions with the query. At least one word in the query have to appear in the content of a question to retrieve the question as a search result.

Step 3: Ranking

Most of the time, the search result contains lots of questions. Ranking results sorts the question in the most relevant order. Most relevant results are to be displayed first. Determining relevancy can be done by various algorithms. Examples of ranking algorithms are Okapi bm25 and PageRank by Google [7]. There may be many questions in the search results. However, normally users are only interested in few of them which are the most relevant. Therefore, ranking is used to ensure that the most relevant questions are returned first [1].

An algorithm chosen for Bugcoli is Tf-idf. This algorithm computes a cosine similarity value between each question contents and a query. Higher cosine values will be assumed that the question is more relevant than others. All questions in the search result are sorted in a descending order according to its computed cosine similarity value. After the user has entered a query, the most relevant results are to be displayed to the user first. Tf-Idf values

do not change according the query, but depends on the modification of the question contents [1]. Therefore, Bugcoli computes the Tf-Idf value during the indexing step, every time new questions are added, as follows:

Step 1: Term Frequency (tf) This procedure is a document-wise statistic. It estimates relevancy of each question in the search by counting occurrences of each word in the question regardless of the order of the words. There is a field in the inverted index called "tf" to store this value.

First, read each word individually in the question's content. Then, if the word is never read before, count its total occurrences and store it into a frequency list. Skip the word that has already been found and move on. When all the words are read, square each value in the frequency list and sum them up together. Then, take a square root the of sum to obtain Euclidean Normal Value. After that, divide each item in the frequency list by the Euclidean Normal Value to obtain a Term Frequency Value (tf). Finally, store the "tf" value in the "Inverted Index" collection, where the term and question's identification corresponds. In conclusion, use the formula for computing the value is written below as follow. Then, store it in the inverted index collection.

$$TermFrequency_i = \frac{f_i}{EuclideanNormal}$$

where

$$EuclideanNormal = \sqrt{\sum_{i=1}^{n} f_i}$$

or

$$TermFrequency_i = \frac{TotalOccurence_i}{\sqrt{\sum_{i=1}^{n} f_i}}$$

where

- n is total distinct words extracted from the content.
- f is the frequency of each term.

Occurrences of each word in a question must be normalized, since questions have its own content length. There are higher possibilities that longer questions will contain higher frequencies of matched terms than shorter questions. However, it does not guarantee that longer questions is more detailed and relevant. This discriminate shorter questions. Therefore, Euclidean Normal eliminates question length discrimination [8].

Step 2: Inverse Document Frequency (idf) Term Frequency considers all words to have equal priority. In other words, all terms in search results have the same weight. Therefore, ranking results with only term frequency technique alone will be priority discriminative, because some words appear few in each question, but may give more specific meaning. While some words appear more, but may be just a common word for grammar purposes with no technical meaning. This results in inaccurate search results.

Therefore, Inverse Document Frequency is a collection-wise statistics. It does not weight terms according to the frequency of the term. Instead, it weights the term according to the amount of questions containing the term. The more questions contains the term, the more common it is. Then, the term is considered more common (not specific) and weights less. For each word in a question, this value counts total questions containing each word regardless of its occurrences. Then, it applies a logarithm with a base of 2 to a fraction of total questions in the website and total questions containing the word, to obtain an Inverse Document Frequency Value of each term as describe in the formula below [8]:

$$InverseDocumentFrequency_i = \log_2 \frac{TotalQuestions}{Document_i}$$

In Bugcoli, there is another collection in the database to be created called "Term_Weight". The collection stores unique words. Each word is linked to its own Inverse Document Frequency Value, and total questions containing the word. Whenever a new question is posted to the website, finding Inverse Document Frequency Value shall be computed after finding the Term Frequency [8].

First, if the word does not exist in the term weight collection, create a new document and set total questions containing the word to one. If the word

is already in the collection, then simply update its total questions value by adding one to the existing value. Then, calculate the Inverse Document Frequency Value by using the formula mentioned above. Under calculation steps of this technique, there is no possible way to obtain negative results [7].

Step 3: Terms Frequency —Inverse Document Frequency Scoring (Tf-idf) Tf-idf scoring is a multiplication of normalizing term frequency and performing inverse document frequency. Tf-idf determines a word frequency of a certain question compared to the inverse proportion of the word over every question in the website. It is considered that a common word in only certain questions tend to have higher Tf-idf value than grammatic words [8].

In Bugcoli, this value is stored in each document of Inverted Index, as an extra field next to the Term Frequency Value. After the values in the previous two steps are calculated, they are multiplied together and stored in this extra field. The formula for calculating this value is as follow [8]:

$$Tf\text{-}idf = TermsFrequency_{i,j} \times InverseDocumentFrequency_i$$

where

- i denotes the word.
- j denotes the question.

Step 4: Cosine Similarity is a value determining the similarity between questions in the search result and the query [6]. This value is used for sorting the search result based on most relevant questions to the query.

In Bugcoli, whenever a user query is received and the search result is not empty, follow the steps below:

1. For the query, all idf values of each word will also be arranged into another vector.
2. For each question, all Tf-idf values of each word in the question which also appears in the query will be arranged into a vector

3. For both the query and each question vectors, calculate a cosine similarity formula to obtain the relevancy value. The formula is written as follows:

$$CosineSimilarity(Query, Question) = \frac{Query \cdot Question}{|Query||Question|}$$

where

$$Query \cdot Question = \sum_{i=1}^{n} Query_i \cdot Question_i$$

and

$$|Query| = \sqrt{\sum_{i=1}^{n} Query_i^2}$$

and

$$|Question| = \sqrt{\sum_{i=1}^{n} Question_i^2}$$

[6] where:

- Query represents a vector of Inverse Document Frequency Values of each distinct word in the query.
- Question represents a vector of Tf-idf Values of each distinct word in the question which is also in the query.
- n represents total distinct words in the query.
- i being subscripted represents the value in the vector which corresponds to a word that both matches in the Query and Question vector.

4. Attach each similarity value is to its corresponding question.

5. This is to be done for each question in search results [8].

6. Sort all the questions in descending order according to the Cosine Similarity Value.

Step four does not operate when new questions are added like the first three step. Instead, step four is only to be performed whenever a user query is obtained. However, in order to gather all information for performing step four, the first three steps must be performed since the question has been added in order to provide some questions to be retrieved.

In conclusion, whenever a question is added to the website, its contents including the question's title, tags, and content are indexed. In addition, the first three steps of ranking are performed. Then, if a user enters a query to search for a question, all questions with at least one word matching any word in the query is retrieved and added to a list known as search results. If the search result is empty, then there are no matches with the query. The search function terminates and awaits a new query. In contrary, each question in the search result is then converted into a vector along with the query. Next, each of the vector, except the query vector, is computed against the query vector to obtain the Cosine Similarity Value. Finally, the search result is sorted according to the most relevant question (Highest Cosine Similarity Value). After all the search and ranking operations are completed, the search result is ready for the server to arrange all information in a web-page for displaying to the user.

3.3.2 Tokenization

Tokenization

Tokenization is a process of segmenting a piece of text into a list of words or assign correct word boundaries on a text. The challenge of Bugcoli is to provide a search engine which can search in Thai. As mentioned before, a text consisting of strings of words are provided all the time, but search functions performs the text as list of words. Therefore, the text must be tokenized to achieve a list of words. However, Thai language is an unsegmented language. The words are written continuously without any delimeters between words. On the other hand, some languages has a boundary between words such as spaces, which is observable [5].

Thai LexTo

Thai LexTo is a tokenizer which can tokenize Thai language text into words. It is available for free by Thailand's National Electronics and Computer Technology Center (NECTEC) through "Sansarn" (http://www.sansarn.com/), known as "LexTo". Accroding to the LexTo's official website (http://www.sansarn.com/lexto), this tokenizer uses Longest Matching Dictionary-Based approach to tokenize Thai texts [5].

Longest Matching dictionary-based approach tokenizes words according to the longest matching of portion of text with the dictionary. This approach considers that longer words have a more specific meaning (higher semantic) than shorter words. The matching concept compares a longest portion of text which also matches a word in a provided dictionary. To compare a portion of text with a list of words in the dictionary, a trie structure is required to store all words in the dictionary [5].

Trie is a tree-like data structure with with multiple children from a parent (Array Tree). Each node stores a single character and has an array to children which also stores other characters. Its name is derived from the word re**trie**val, in information retrieval systems by Fredkin [1960] [3].

Thai LexTo uses a trie data structure to store all words from the dictionary. When traversing the trie from the parent and append the characters to form a word along the path, if any node makes the word exist in the dictionary, that node is marked as a valid word. Similar words with matching prefixes is only stored once up until the last character in the matching prefix. Then, that last node branches out to store other words. During words matching, trie allows words with similar prefix to skip comparing the same characters to the character behind the prefix. When LexTo tokenizes, it matches the text with the deepest node which is marked as a valid word. Then, it marks in the text as a position to segment the word and reads the next character as a first character of the next word [5].

In conclusion, LexTo is Thai language tokenizer for segmenting Thai words from any provided sentence. It plays the crucial, as Bugcoli search performs on a list of words, not a single piece of text. It uses Longest Matching

approach, which requires a dictionary file to be provided for storing all words to be compared in a trie for matching the words in the dictionary with a text from the website.

Chapter 4

Methodology

4.1 Development Methodology

The development process being used is Kanban. Kanban is a framework used to implement and elaborate agile process for managing knowledge work with an emphasis on just-in-time delivery, without overloading development team members. This methodology is used to visualize the flow of program using Kanban Board, by assigning lists of To-do, Doing, and Done tasks. Each group member will be able to see the progress each task without waiting for others to finish other tasks. This increases the quality of each deliverable task of each member and prioritizes the task by putting the highest important first. This approach presents all participants with a full view of the process from task definition to the delivery of the project

4.2 Requirements

User Requirements: Functional

- The system allows users to register and receive membership from the website. Member can log-in to the website through the created account, or via social media such as Facebook, Twitter, Instagram, and Google+.
- Membership allows users to post questions.
- Membership allows users to give a vote to questions.
- Membership allows users to attach tags according to the topic of the question while writing the new question.
- Membership allows users to give a vote to answers.
- Membership allows users to answer questions.
- Membership allows users to bookmark questions.
- Bookmarked questions can be quickly accessed by the member, despite being difficult to find for other members. (Marking as Favorite)
- The owner of each question is able to select the best answer for each question, and the owner of the selected answer will receive points.
- The system provides a "Report" feature on every question, answer, and answer replies for all users, if the content of the question, answer, or answer contains inappropriate contents or questions which are unrelated to tags, or answers and answers which are related to the question.
- The system allows members to view website usage statistics:
  - User's Current Rank
  - Current Points Obtained
  - Total Questions Asked
  - Total Answers Answered
  - Total Questions Voted

  - Total Answer Answers Given

- The system allows users to customize account settings, and user information.
- The system allows users to search for questions.
- Members are able to view their own notifications, and be able to view the web-page where notification happens.

User Requirement: Non-Functional

- Members will receive notifications on the right top of website, under the following circumstances:
  - Other members answered questions written by the owner.
  - Other members voted for the question written by the owner.
  - Other members voted for the answer.
  - Any question owner has selected the answer their best answer.
- Members will receive error notifications for voting or selecting their own answer as the best answer.
- A web-page design must contain header of top of every page.
- A head must contain a website logo, search box, and little main menu
- The system must respond to each users' request within 5 seconds.
- Users can obtain points after receiving votes for their questions or answers from other website members.
- All users can search for questions in Thai and English language.

System requirement: functional

- Database: The website stores its data on MongoDB.
  - The database will be directly accessed by certain methods on the server side.
  - The database consists of multiple collections of multiple documents.
  - Documents on the database is stored in JSON format.
- Points:
  - Members obtain 5 points per vote received on their questions or answers.
  - Members get 5 points deducted for each disagree received.
  - Points within certain range will assign each members certain ranks.
  - Ranks help to determine member's reliability and reputation.
- Ranks: Members' ranks will be promoted after receiving up to certain points
- Members' ranks indicate how much they have contributed to the Bugcoli's community.
- Promoting ranks increases the members' reliability and reputation in the community.
- Webboard: Main component of the system:
  - Displays a list of questions in a well-decorated format.
    * Question title (Top-left position)
    * Tags of the question (Position below question title)
    * Time posted (Bottom-right position below the question detail)
    * Owner of the question (To the right of time posted, begins with a word "by")

    * Bookmark Button (Top-right button, only displayed when logged in)
    * Statistics Panel (Entire left panel next to the question). Displays the following information about the question from left to right respectively:
        · Votes —Total votes received from other community members.
        · Disagrees —Total disagrees received from other community members.
        · Answers —Total answers in the question.
        · Views —Total visitors and members who accessed to the question.
  - Questions must be partly displayed
  - Each question must contain a link to a page displaying its complete content.
  - The system allows users to view questions on a webboard.

System Requirements: Non-Functional

- The website has Secure Sockets Layer (SSL) protection.
- The client side frame work is developed using ReactJS.
- The server side is developed using GO language.
- The website must be viewable and accessible with any browser on any device.
- The website must be deployed in Amazon cloud service.

4.3 System analysis and design

4.3.1 Use case and description

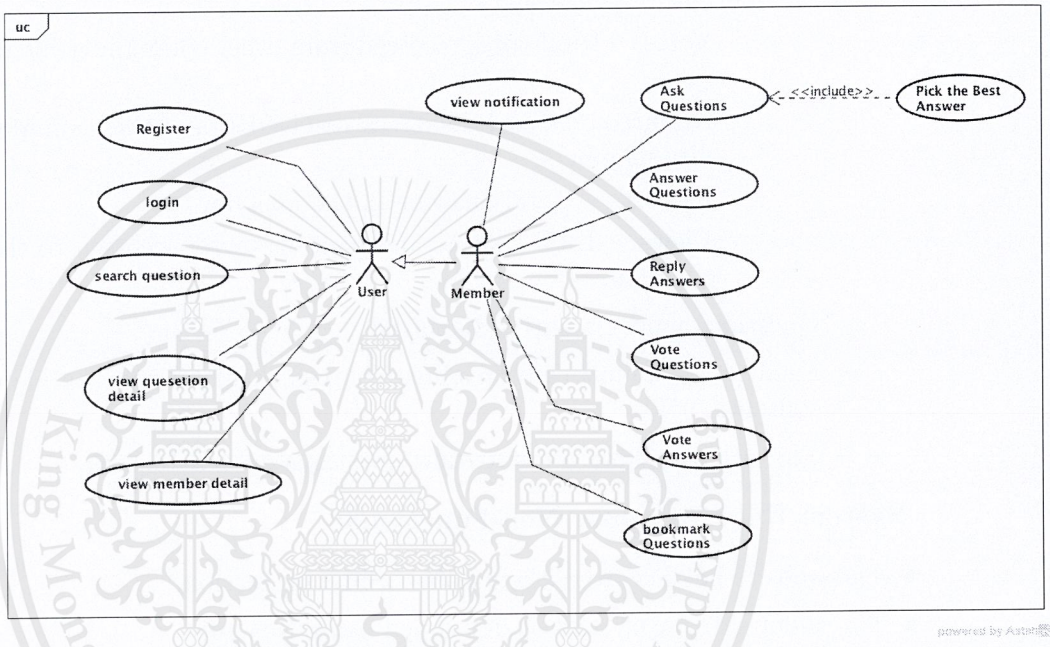


Figure 4.1: Use-Case Diagram

Use-Case #1	Register		
Primary Actor	General User		
Precondition	The user is not logged in		
Postcondition (Success)	The user's entered information is added to the website's database		
	The user can login to the website		
	New users will not be able to register with this information again		
Postcondition (Failed)	User's entered information is removed, go back to Homepage		
Trigger	The user selects the link to register page		
Flow of Events	Step	Actor's Input	System's Response
	1		Display a registration page.
	2	Fill in all information: - Name - Email - Password - Profile Image	
	3	Check if user's information exists.	
	4	Update user's information to the database.	
	5	Go back to homepage.	
Alternate Events	Step	Actor's Input	System's Response
	3		Check if user's information exists.
	3.1		User's information exists.
	3.2		Redirect to login page.
	3.3		Ask the user to login.
	3.4	User authenticates correctly.	
	3.4.1		Login success.
	3.5	User authenticates incorrectly.	
	3.5.1		Display an error.
	3.5.2		Ask the user to login again.
	3.5.3		Repeat step 3.2.
	3.6	User does not login.	
	3.6.1		Go back to homepage.

Table 4.1: Register Use-Case Description

Use-Case #2	Search		
Primary Actor	General User and Members		
Precondition	The user has entered a query		
Postcondition (Success)	A list of results is displayed		
	A query is cleared		
Postcondition (Failed)	No results found		
Trigger	The user clicks a search button		
Flow of Events	**Step**	Actor's Input	System's Response
	1	Enter a query.	
	2		Fetch results.
	3		Display results.
Alternate Events	**Step**	Actor's Input	System's Response
	2		Fetch results.
	2.1		No results match the query.
	2.2		Display no results.

Table 4.2: Search Use-Case Description

Use-Case #3	View Question in Details		
Primary Actor	General User and Members		
Precondition	The user is browsing a list of questions		
Postcondition (Success)	Go to a webpage to view the question		
Trigger	The user selects the question link		
Flow of Events	**Step**	Actor's Input	System's Response
	1	Click on the link to question.	
	2		Get the link's unique id.
	3		Fetch question's contents.
	4		Arrange contents and display.

Table 4.3: View Question in Details Use-Case Description

Use-Case #4	Login with an application		
Primary actor	Members		
Precondition	User must have a membership		
Postcondition (Success)	User is authenticated		
Trigger	Log-in successful		
Flow of events	The user has accessed an application to login		
	Step	Actor's Input	System's Response
	1	User wants to login.	
	2	User selects an application on the provided list:	
		- Facebook - Twitter - Google Plus - LinkedIn - Github - Bugcoli Account	
	3	Validate username and password.	
	4	Display welcome if validate login successful or Display invalid login if validate login fail.	
	5	Click confirm button.	

Table 4.4: Login with an Application Use-Case Description

Use-Case #5	View Question in Details		
Primary actor	General User and Members		
Precondition	The user is browsing a list of questions		
Postcondition(Success)	Go to a webpage to view the question		
Trigger	The user selects the question link		
Flow of events	**Step**	Actor's Input	System's Response
	1	Click on the link to question.	
	2		Get the link's unique id.
	3		Fetch question's contents.
	4		Arrange contents and display.

Table 4.5: View Question in Details Use-Case Description

Use-Case #6	View Members in Details		
Primary actor	General User and Members		
Precondition	The user is browsing a list of questions		
Postcondition(Success)	Go to the member's profile page		
Trigger	The user selects the link to the member's profile		
Flow of events	Step	Actor's Input	System's Response
	1	Click on the link to the profile.	
	2		Get the link's unique id.
	3		Fetch that member's contents.
	4		Arrange contents and display.

Table 4.6: View Members in Details Use-Case Description

Use-Case #7	Ask a Question		
Primary Actor	Members		
Precondition	The user is logged in		
Postcondition(Success)	The question is posted webboard		
Postcondition (Failed)	The system asks the user to login		
Trigger	The user selects the link to write new question		
Flow of Events	Step	Actor's Input	System's Response
	1	Click on a button to write new question.	
	2		Receive inputs.
	3		Store the question to database.
	4		Post the question on webboard.
Alternate Events	Step	Actor's Input	System's Response
	1	Click on a button to write new question.	
	1.1		User is not logged in.
	1.2		Redirect to login page.

Table 4.7: Ask a Question Use-Case Description

Use-Case #8	Pick a best Answer		
Primary Actor	Members		
Precondition	The user is logged in		
	The user asked the question		
Postcondition(Success)	The user who answered obtains a score		
Postcondition (Failed)	The system asks the user to login		
Trigger	The user who asked the question presses the best answer		
Flow of Events	Step	Actor's Input	System's Response
	1	Click on an answer to vote.	
	2		Mark the answer as the best.
	3		Grant the answerer some scores.
Alternate Events	Step	Actor's Input	System's Response
	1	Click on an answer to vote.	
	1.1		User is not logged in.
	1.2		Redirect to login page.

Table 4.8: Pick a best Answer Use-Case Description

Use-Case #9	Answer a Question		
Primary Actor	Members		
Precondition	The user is logged in		
	The user did not ask the question		
Postcondition(Success)	The answer is posted to the question		
Postcondition (Failed)	The system asks the user to login		
Trigger	The user is viewing the question in details		
Flow of Events	**Step**	Actor's Input	System's Response
	1	Enter an answer.	
	2		Update the answer to database.
	3		Post the answer to the question.
Alternate Events	**Step**	Actor's Input	System's Response
	1	Enter an answer.	
	1.1		User is not logged in.
	1.2		Redirect to login page.

Table 4.9: Answer a Question Use-Case Description

Use-Case #10	Reply to an Answer		
Primary Actor	Members		
Precondition	The user is logged in		
	The user is viewing the question in details		
Postcondition(Success)	The reply is posted to the answer		
Postcondition (Failed)	The system asks the user to login		
Trigger	There exists an answer to reply		
Flow of Events	**Step**	Actor's Input	System's Response
	1	Enter a comment.	
	2		Update the answer's comment to database.
	3		Post the reply to the answer.
Alternate Events	**Step**	Actor's Input	System's Response
	1	Enter a comment.	
	1.1		User is not logged in.
	1.2		Redirect to login page.

Table 4.10: Reply to an Answer Use-Case Description

Use-Case #11	Vote a Question		
Primary Actor	Members		
Precondition	The user is logged in		
	The user is viewing the question in details		
	The user has never voted the question		
Postcondition (Success)	The question gains a rating		
	The user gains some points		
Postcondition (Failed)	The system asks the user to login, or to devote the question		
Trigger	The user presses the vote button on a question		
Flow of Events	**Step**	Actor's Input	System's Response
	1	Click on a button to vote.	
	2		Update the question's score in database.
	3		Update the user's score in database.
Alternate Events	**Step**	Actor's Input	System's Response
	1	Click on a button to vote.	
	1.1		User is not logged in.
	1.2		Redirect to login page.
Alternate Events	**Step**	Actor's Input	System's Response
	1	Click on a button to vote.	
	1.3		The user has already voted the question.
	1.4		Ask the user whether to devote.
	1.5	The user devotes.	
	1.5.1		Update the question's and user's score in database.
	1.6	The user does not devote.	
	1.6.1		Nothing changes.

Table 4.11: Vote a Question Use-Case Description

Use-Case #12	Vote an Answer		
Primary Actor	Members		
Precondition	The user is logged in		
	The user is viewing the question in details		
	The user has never voted the answer		
Postcondition (Success)	The answer gains a rating		
	The user gains some points		
Postcondition (Failed)	The system asks the user to login, or to devote the answer		
Trigger	The user presses the vote button on an answer		
Flow of Events	**Step**	Actor's Input	System's Response
	1	Click on an answer to vote.	
	2		Update the answer's score in database.
	3		Update the user's score in database.
Alternate Events	**Step**	Actor's Input	System's Response
	1	Click on an answer to vote.	
	1.1		User is not logged in.
	1.2		Redirect to login page.
Alternate Events	**Step**	Actor's Input	System's Response
	1	Click on an answer to vote.	
	1.3		The user has already voted the answer.
	1.4		Ask the user whether to devote.
	1.5	The user devotes.	
	1.5.1		Update the answer's and user's score in database.
	1.6	The user does not devote.	
	1.6.1		Nothing changes.

Table 4.12: Vote an Answer Use-Case Description

Use-Case #13	Bookmark a Question		
Primary Actor	Members		
Precondition	The user is logged in		
	The user is viewing the question in details		
	The user has never bookmarked the question before		
Postcondition (Success)	The user's bookmark list is updated		
Postcondition (Failed)	The system asks the user to login, or to remove the bookmark		
Trigger	The user presses the question's bookmark button		
Flow of Events	**Step**	Actor's Input	System's Response
	1	Click on the bookmark button.	
	2		Update the user's bookmark list in the database.
	3		Mark the question as booked by the user.
Alternate Events	**Step**	Actor's Input	System's Response
	1	Click on the bookmark button.	
	1.1		User is not logged in.
	1.2		Redirect to login page.
Alternate Events	**Step**	Actor's Input	System's Response
	1	Click on the bookmark button.	
	1.3		The user has already bookmarked the question.
	1.4		Ask the user whether to remove the bookmark.
	1.5	The user removes the bookmark.	
	1.5.1		Update the question's and user's score in database.
	1.6	The user does not remove the bookmark.	
	1.6.1		Nothing changes.

Table 4.13: Bookmark a Question Use-Case Description

Use-Case #14	See Notifications		
Primary Actor	Members		
Precondition	The user is logged in		
Postcondition (Success)	The notification is added to user's list as unread		
Trigger	An activity related to the user, caused by other members has occurred		
Flow of Events	**Step**	Actor's Input	System's Response
	1	Other users does an activity.	
	2		Update the event to database.
	3		Display the notification in the user's list.

Table 4.14: See Notifications Use-Case Description

4.4 Data modeling

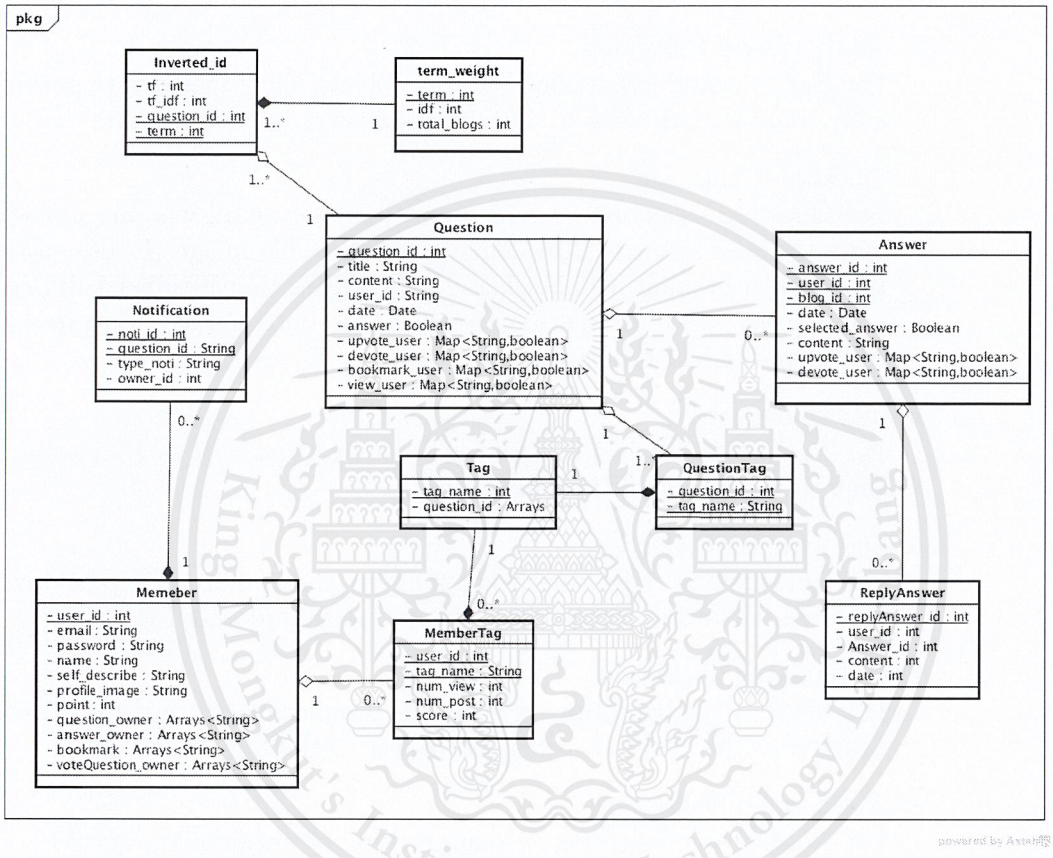


Figure 4.2: Class Diagram for Database Model

1. Question Collection:
   For Question collection, it stores created questions which contains a title, content, and date. It also stores member_id. If any member votes, then it stores upvote_user, devote_user attributes. It stores member_id if member bookmark question in bookmark_user, and store IP visiting in view_user to see how many user visiting this question.

2. Answer Collection:
   For Answer collection, it store answers of questions which contain

content, data , question_id of each question, selected_answer to know whether the answer is the best answer or not. It also stores memer_id if member vote ,then store in upvote_user, devote_user attributes.

3. ReplyAnswer Collection:
   For ReplyAnswer collection, Bugcoli website allow member to specifically reply to each answer. It contains answer_id, content, and date

4. Question Collection:
   For Question collection, it tells about member detail who are already login to the system. it stores email, name, profile_image. It also stores the number of point member has, the questions that member had been created, the questions that member had been bookmarked, the question that member had been voted.

5. Tag Collection:
   For Tag table, it collect unique tag name inside as well as all questions that have been using this current tag

6. QuestionTag Collection:
   For QuestionTag collection, it is a many to many relationship between Question table and Tag table. It store question id and Tag name

7. MemberTag Collection:
   For MemberTag collection, it store the number of tag that member is using when he create, re when he answer to the tag of question.

8. Notification Collection:
   For Notification collection, it contain type of notification, question id and the member that send notification to you.

9. Inverted_Index Collection:
   Inverted_Index collection stores terms attached to its corresponding tf-idf and the identification to the question in the question collection. The relationship between terms and each identification of the question is many-to-many.

10. Term_Weight Collection:
    Term_Weight collection stores distinct terms attached to its corresponding Inverted Document Frequency value. Each document also tracks total questions containing each term.

4.4.1 Package diagram

The package diagram in figure 4.3 present the big picture system design of Bugcoli project. There are website part, server side, database part, and search part. Website part or client side is the user-interaction in web browser of user. Server side is the main operations which provide command for storing data, retrieving data, and process the search algorithm. Search part is the part where questions' indexes are created and retrieved in order to get the most related questions. Database part is to store and retrieve data.

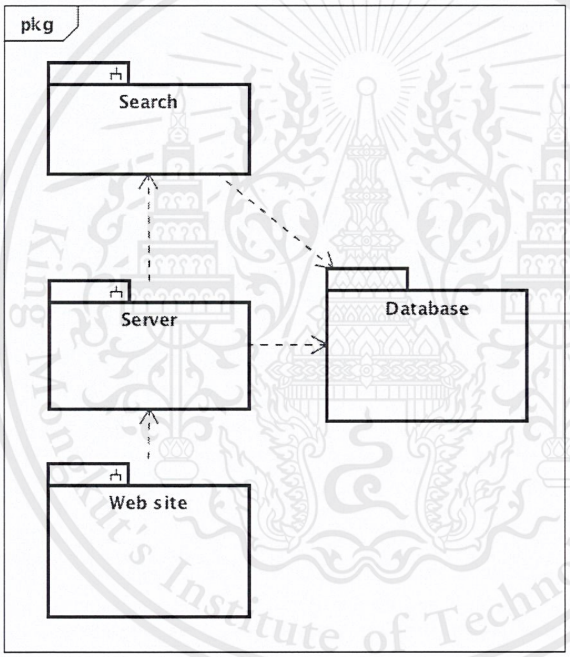


Figure 4.3: Bugcoli Package Diagram

4.4.2 Class diagram

Website Class Diagram

The class diagram in figure 4.5 illustrate the collections of user-features as a method in each class. It also shows the relationship between features and sub-features. For example, reply class is contained in Answer class and Answer class is contained in Question class. User_tag store the user interest tags, stored in MemberProfile (user information) class

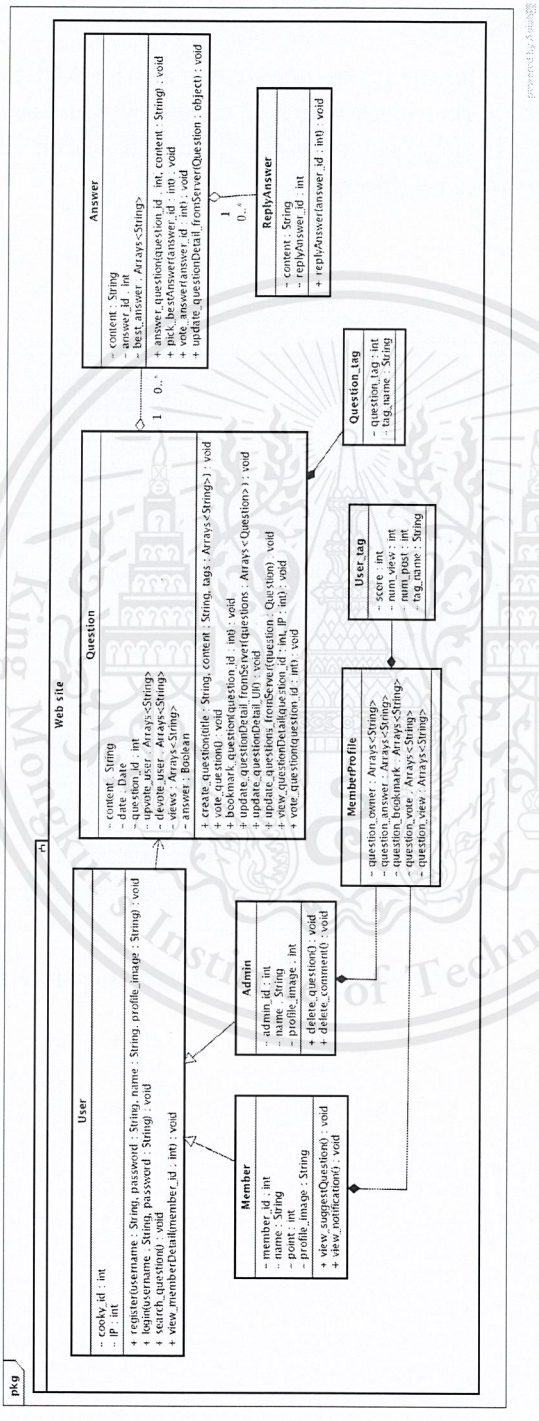


Figure 4.4: Bugcoli's Website Website Class Diagram

Server Class Diagram

The class diagram in figure 4.5 demonstrates all the API for Bugcoli project. Each of method in Server class which is called by website class diagram will call other methods in database class diagram and search class diagram, then send back to website class diagram.

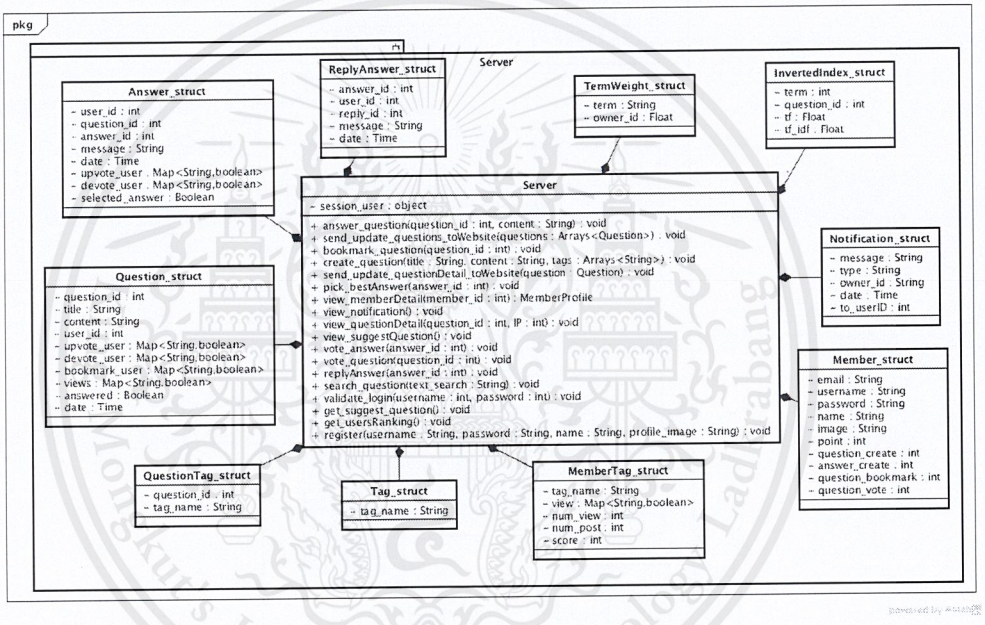


Figure 4.5: Bugcoli's Website Server Class Diagram

Database Class Diagram

The class diagram in figure 4.6 present the methods in order select, insert, update, delete information inside database model.

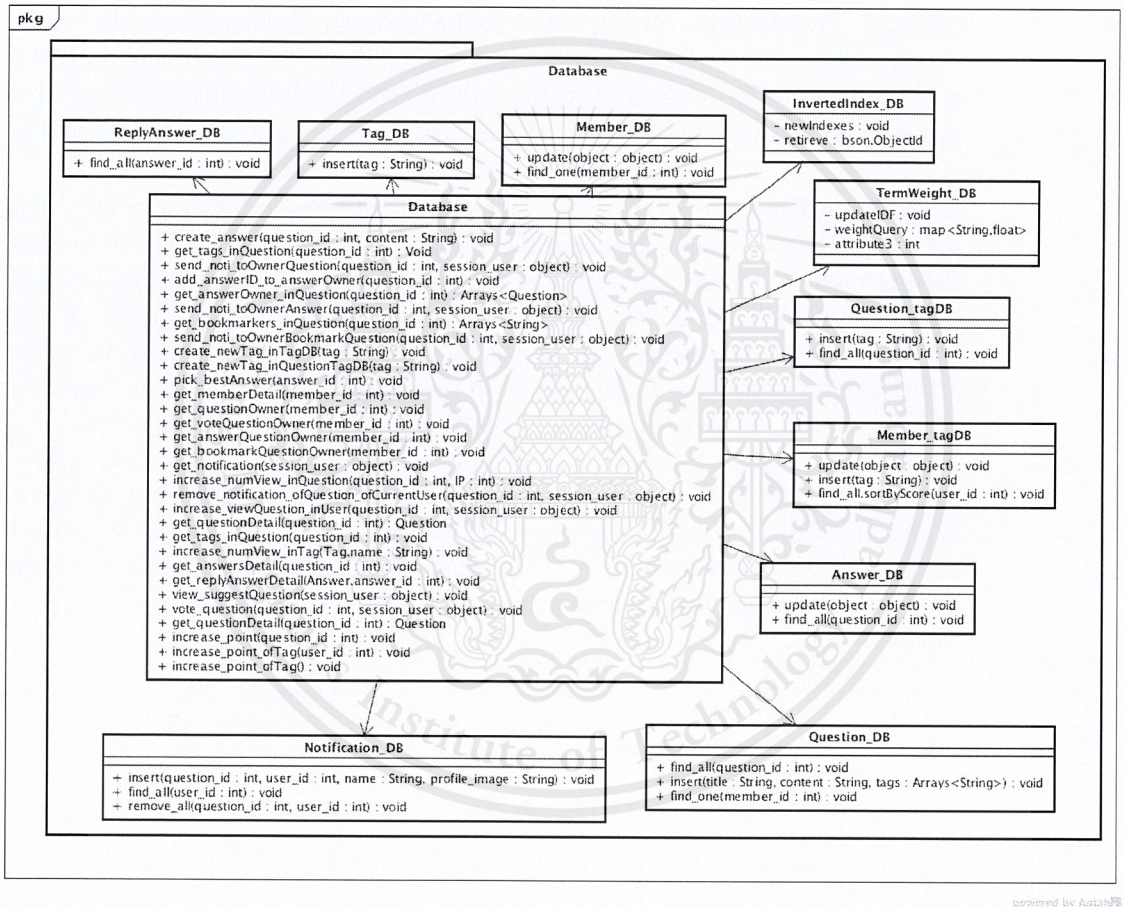


Figure 4.6: Bugcoli's Database Class Diagram

Search Class Diagram

Searcher Class indexes all immediately added questions and performs retrieval and ranking, which are the basics of searching as stated in the third chapter (Background Knowledge - Search Algorithm). Before both indexing and searching event, all texts must be tokenized into terms. Therefore, the searcher class will require a service from "LongLexTo" class in Tokenizer package. In addition, during the ranking process, the searcher will require sorting from a Golang Built-In Package called "sort". The sorting will be in a descending order according to the only floating-point value attached to its corresponding question identification stored in each "Similarity" struct type.

Tokenizer Package tokenizes a Thai text into words. Its main algorithm for tokenizing Thai text into words is Longest Matching. In order to perform the matching algorithm, an input text must be compared with words provided dictionary. In Thai Lexto, a dictionary text file containing a list of only words must be provided. All words in the file must be added to a Trie structure for matching with words in the input text. Longest Matching algorithm will be performed in "LongParseTree" class, which is called in "LongLexTo" class. Since Thai LexTo also handles white spaces and numbers [5], a service from "unicode" class will be required to check the character type. In addition, since Thai LexTo creates a list of integers, which acts as an index to character in a string that marks a position for tokenizing, Tokenizer package will tokenize that into strings and store it in a list of Tokens.

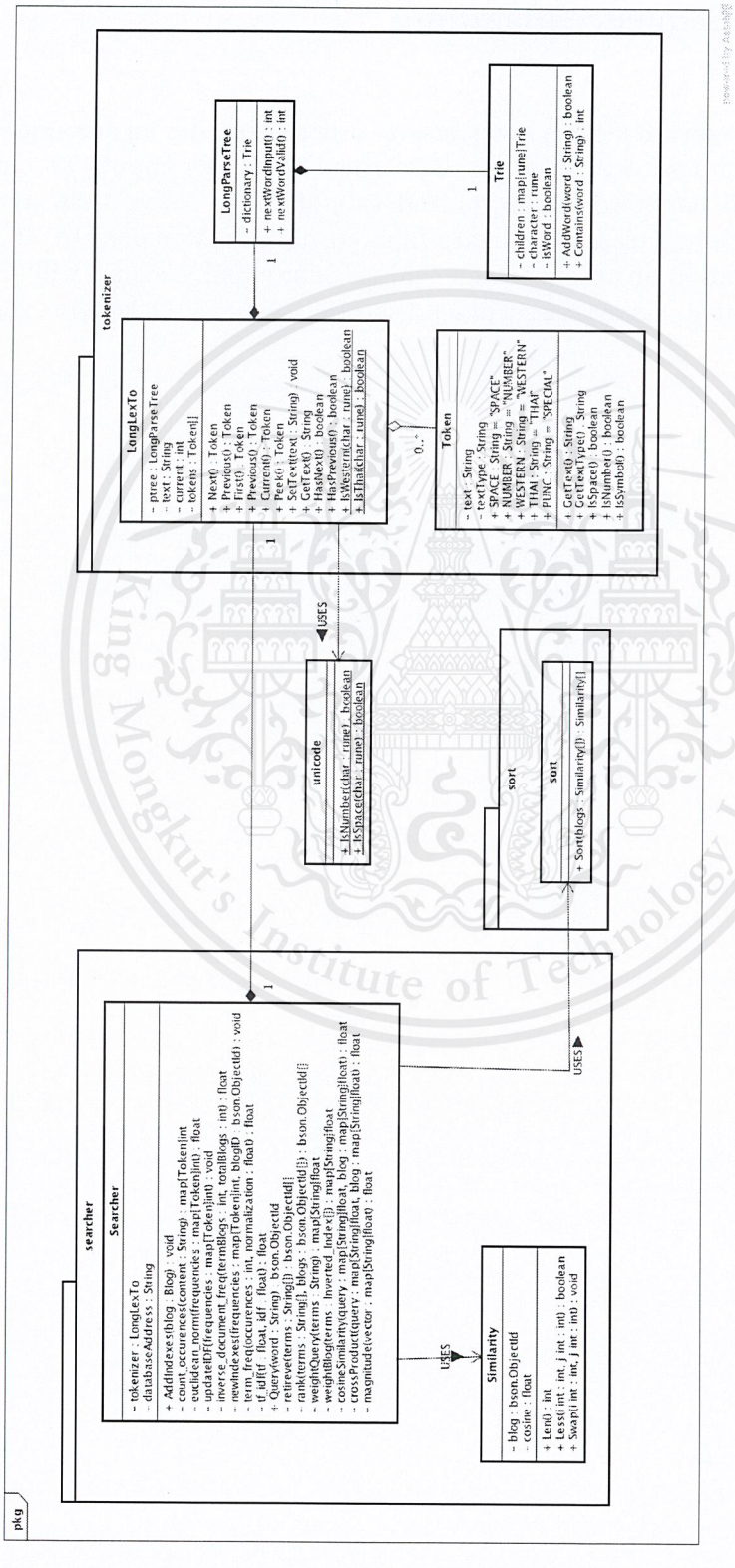
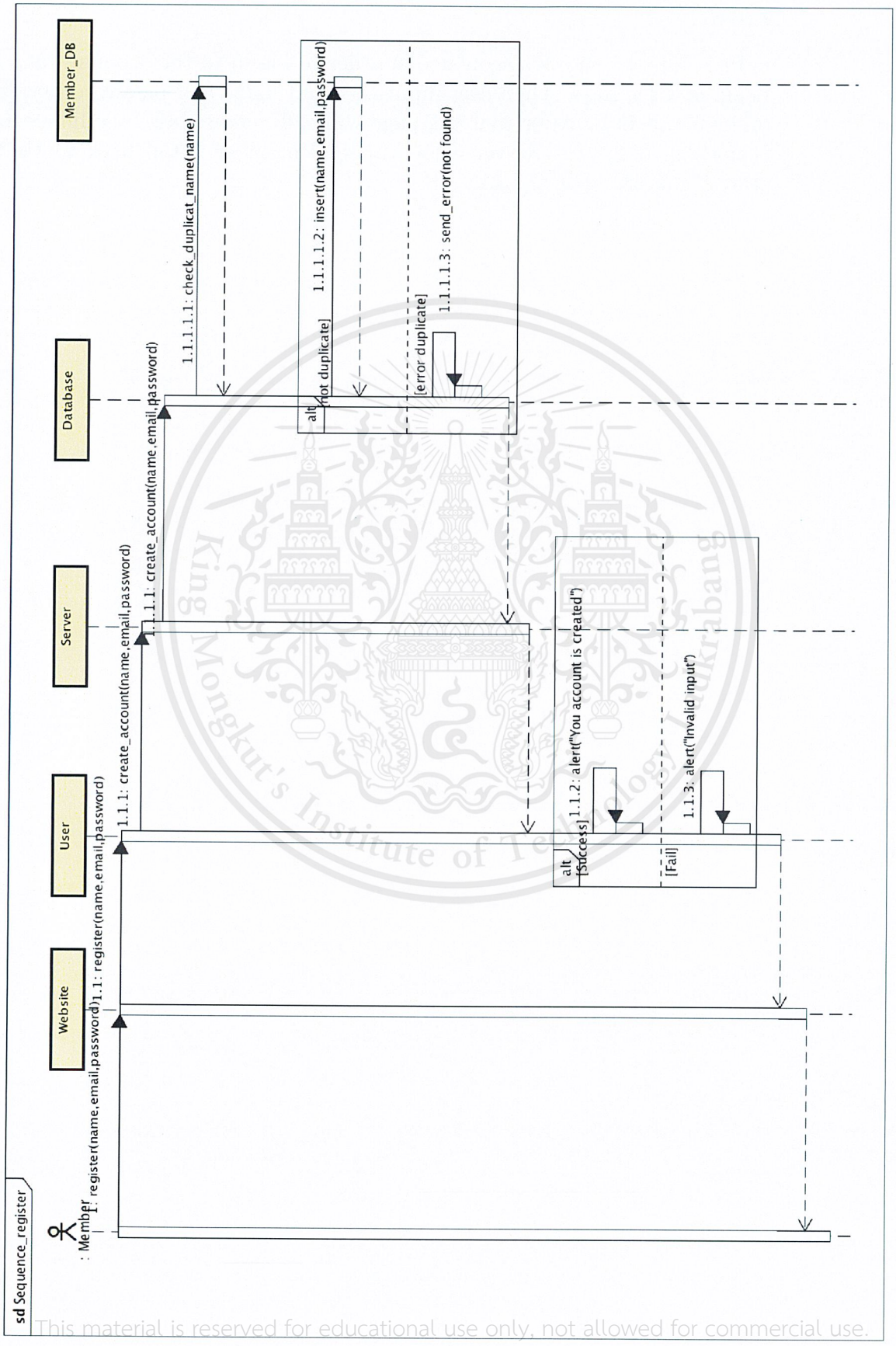


Figure 4.7: Bugcoli's Search Class Diagram

4.4.3 Sequence diagram

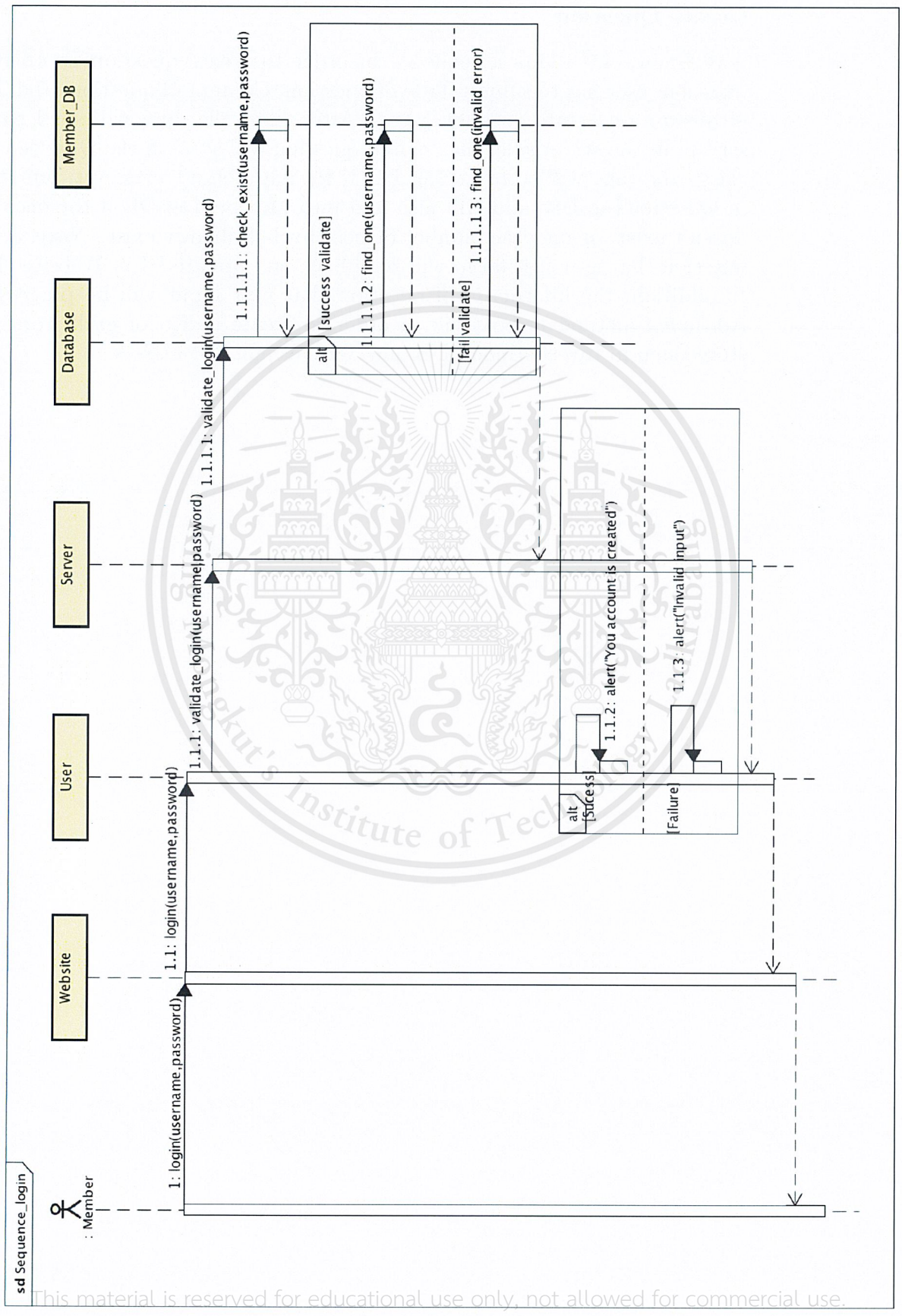
Register

From Figure 4.4 shows a sequence in order to register an account to register an account,user will begin at User class.Then user inputs account name, email and password into form and submit data. After that, the website will call Server class to validate inputed data in Member_DB. If there are no duplication on account name and account email, account will be created. After created, Server class will send succeed message. Otherwise, send failure message.



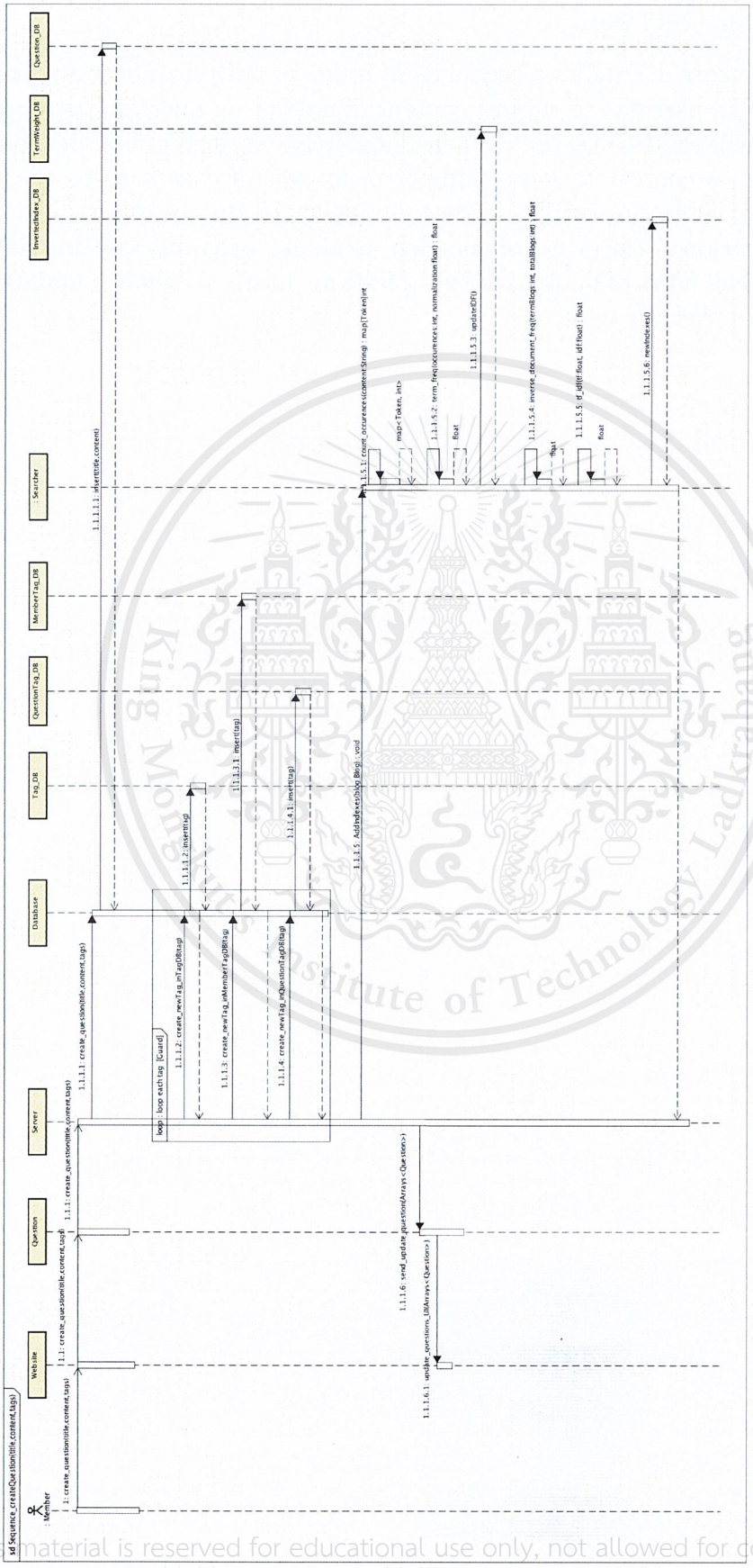
Login

In Figure 4.5 shows a sequence in order to log in to the account, user will begin at User class. Then user inputs account name and password into form and submit data. After that the page will call Server class to validate data. If nothing is invalid, Server class will send member detail back to User to store it and show on website



Create Question

In Figure 4.6 shows a sequence in order to create question.To create a question, user has to submit Title of question, Content of question, and Tags of qustion on create question page. After that, the data will send to the server. In the server side, the created question will store in the Question_DB class. the tags will store in Tag_DB if the tag doesn't exist yet, will store in QuestionTag_DB, and will also add into MemberTag_DB if the each tag doesn't exist, or increase number of post by 1 if already exist. Next, a new question that user just fill in will send back and update UI in Website class. In addition, the informatio of question that user input will be processs in AddIndex method in Searcher in order to create index for each word and store them in InvertedIndex_DB and TermWeight_DB classes



Answer Question

In Figure 4.7 shows a sequence in order to answer questions. To answer a question, user has to submit content of answer on question detail page. The data will send to server. When data arrive at server, first it create a new answer, second it increase number of answer of each tags to the answerer, third it store answerID in Question_owner of the answerer, next it send a notification to the owner of question, to all users that answers in this question, and to all users of bookmark this question. Lastly, it sends a update question page to Website class.

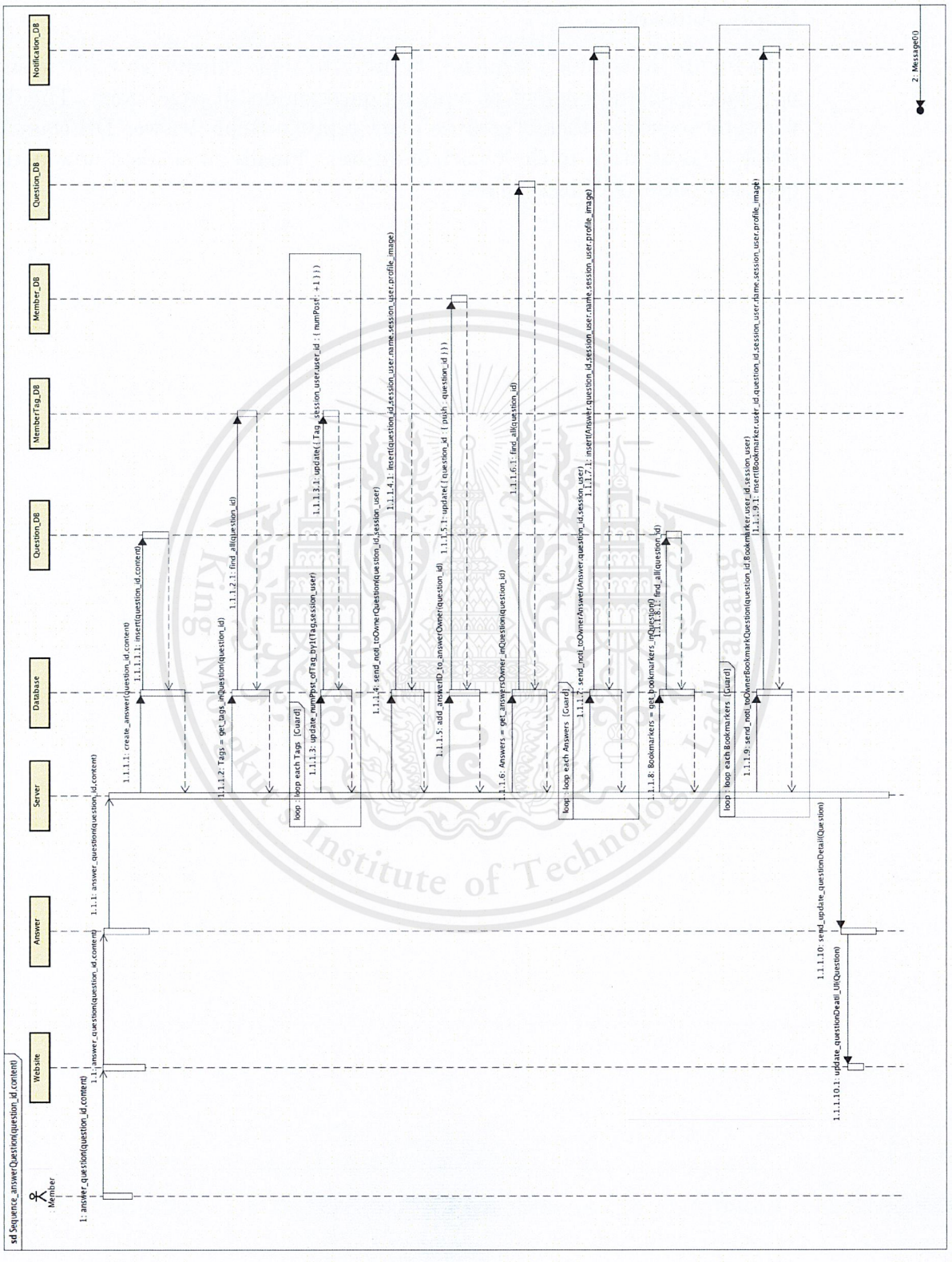
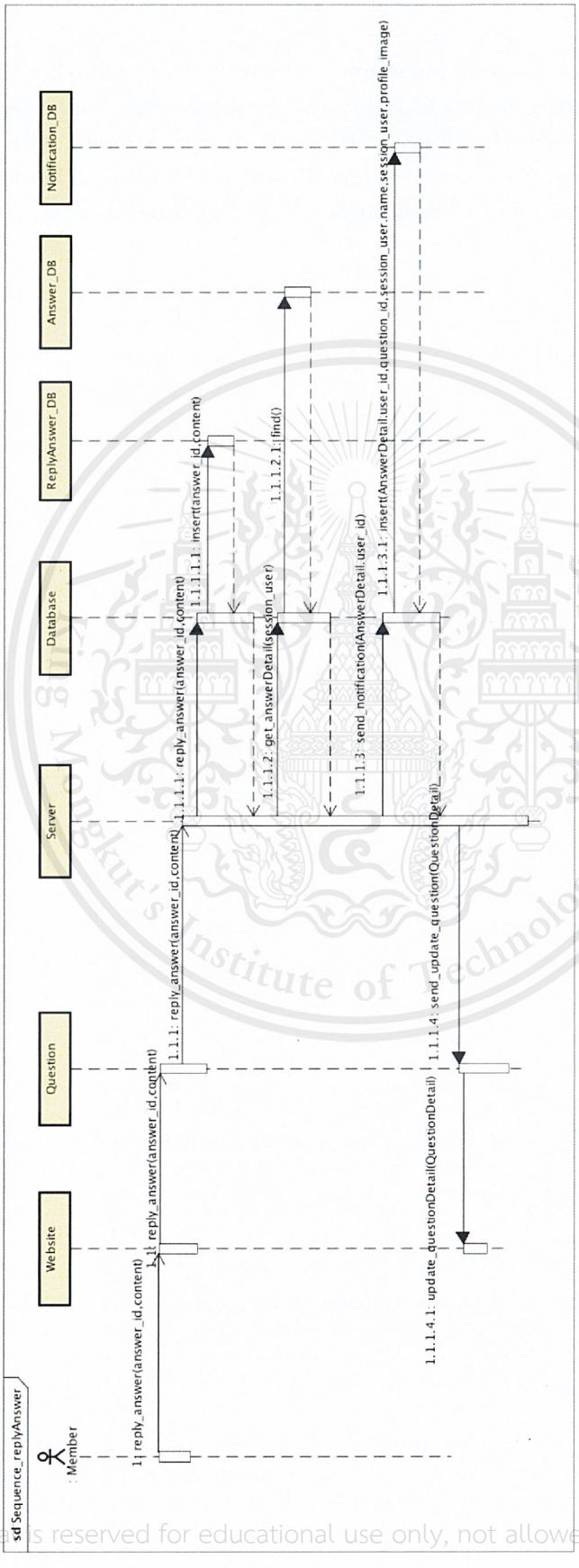


Figure 4.11: Answer Question sequence diagram

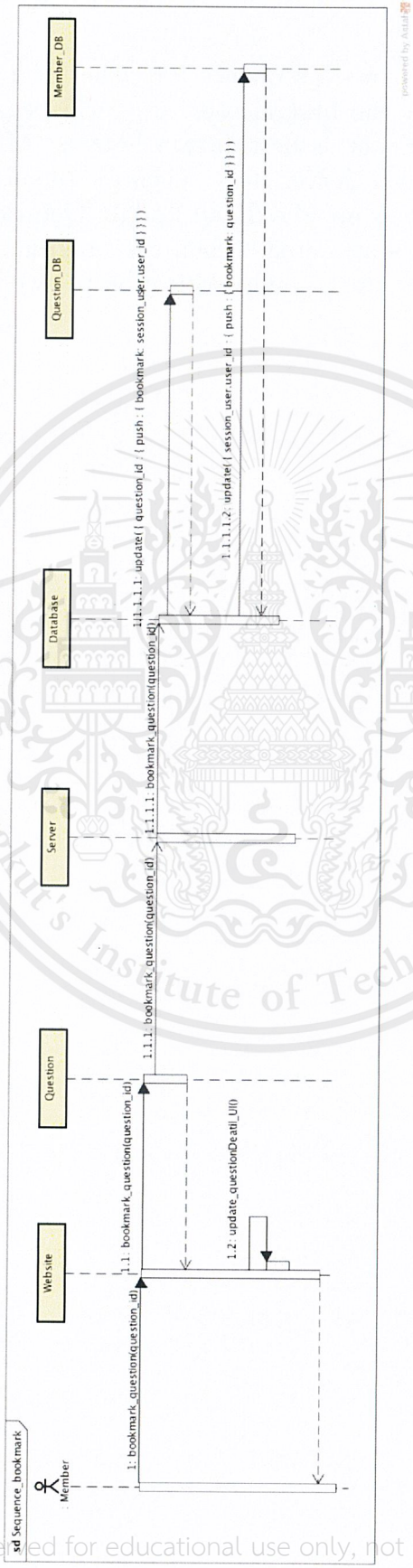
Reply Answer

In Figure 4.8 shows a sequence in order to reply answer.To reply answer, user has to submit content of reply on question detail page. Next, The data will send to server, then it creates a new reply in ReplyAnswer_DB class and sends a notification to the owner of answer. Finally, it sends a new update question page to Website class.



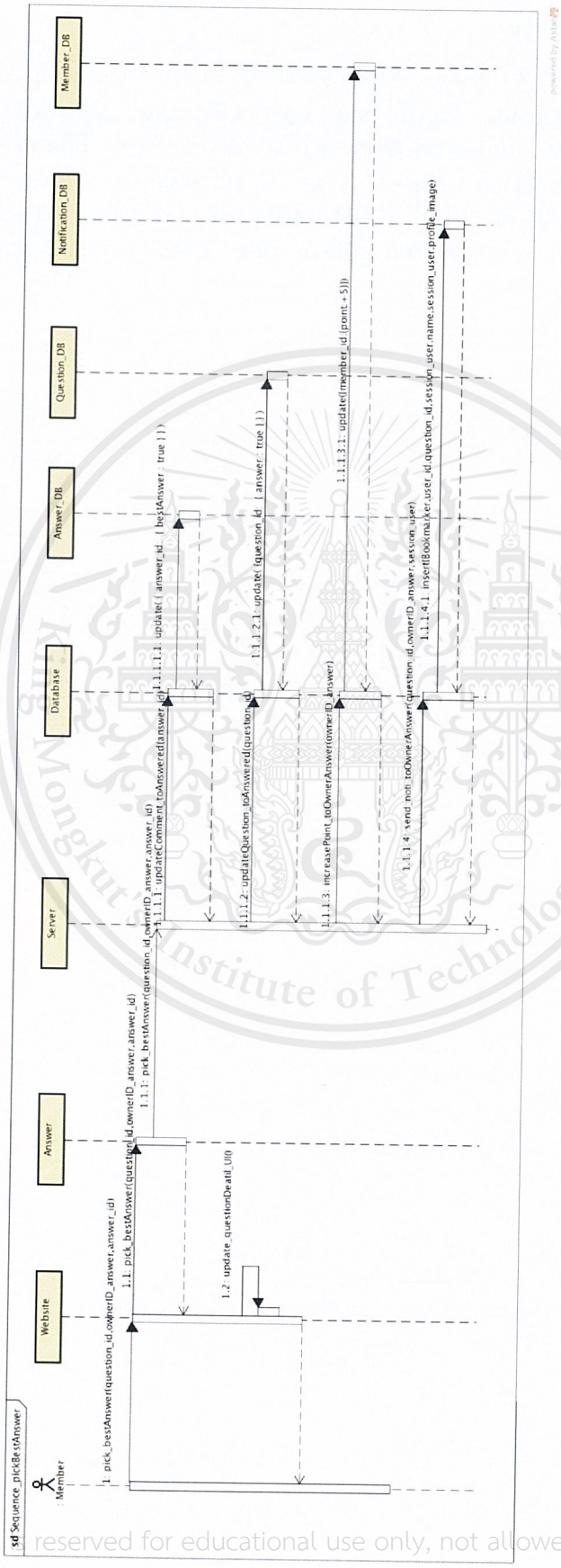
Bookmark

In Figure 4.9 shows a sequence for user to bookmark a question. To bookmark questions, user will begin at Question class, then click bookmark button. After user click bookmark button, question id will be sent to Server class. Then question id will be stored in Member_DB class of bookmarker and stored user id of bookmarker in Question_DB class



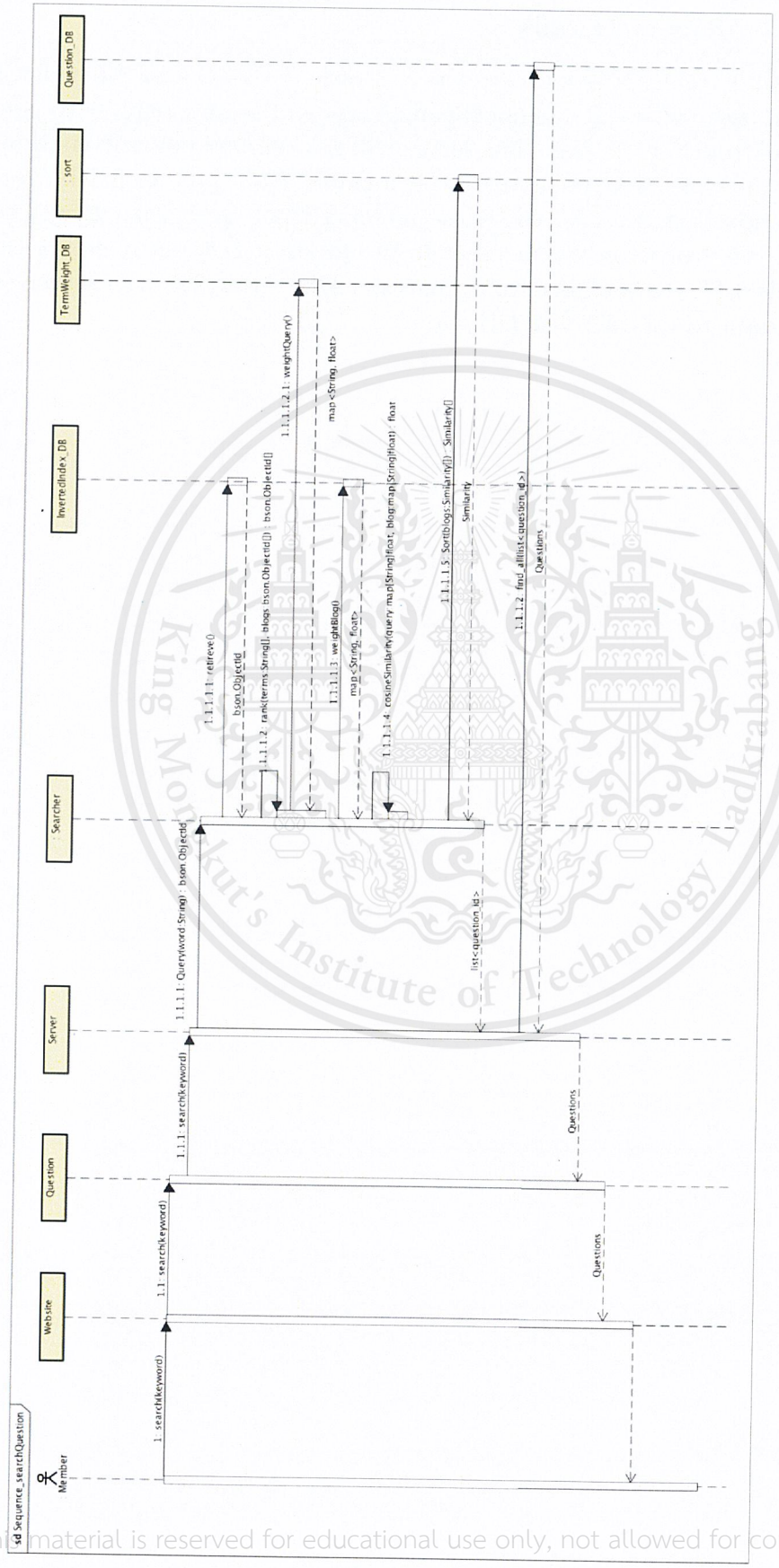
Pick best Answer

In Figure 4.10 shows a sequence for owner of question to select the best answer. To select the best answer, user will begin at Answer class, then click select this answer as best answer button. After user click the button, answer id will send to Server class. Then the server update Answer_DB class of answer id to be as answered and update Question_DB class that contant answer od to be as answered. Then the program increases point of owner of answer by 5. Lastly, it send notification to owner of answer that is being selected.



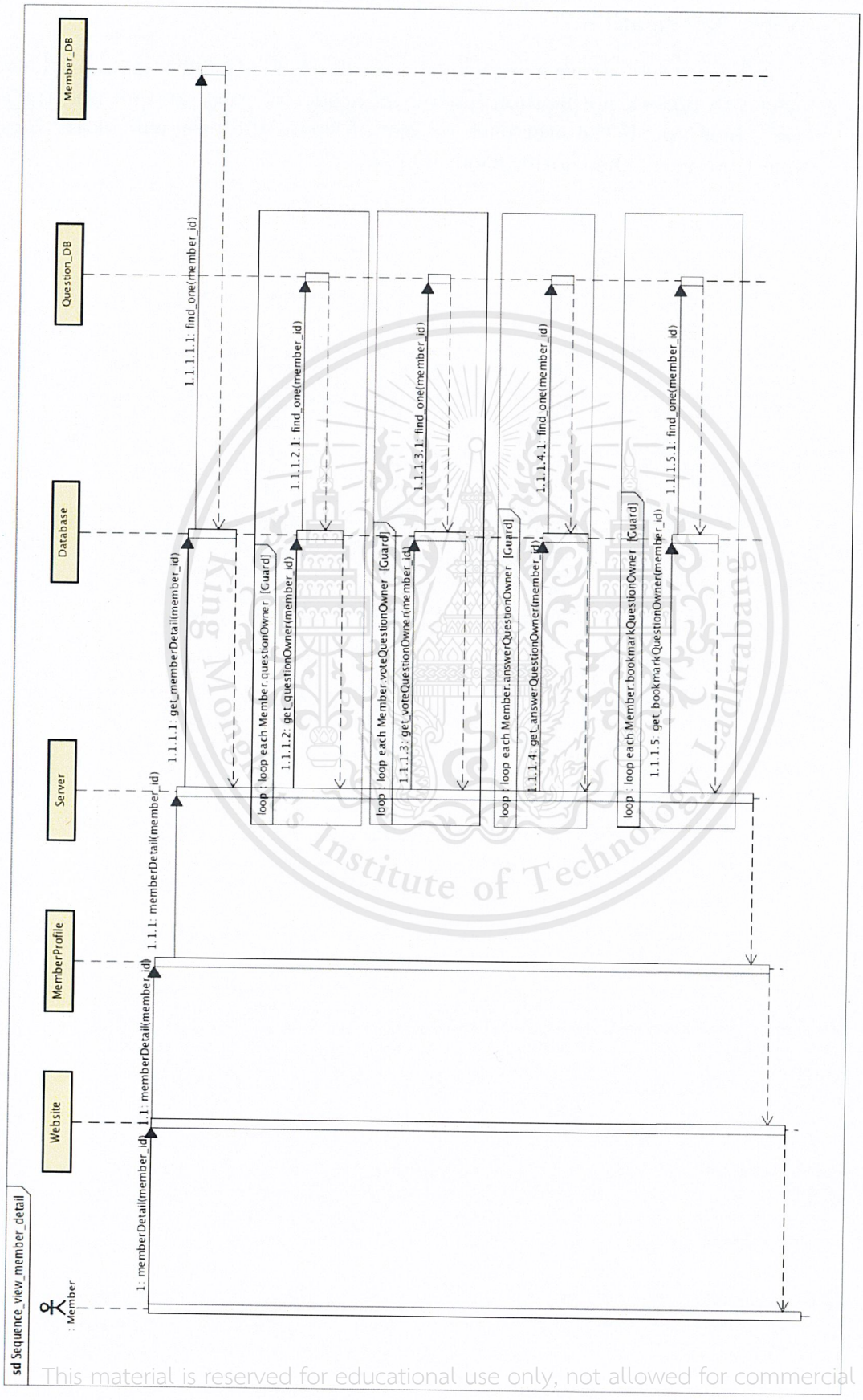
Search Question

In Figure 4.11 shows a sequence in order to search for related question to keyword input. To do that, user will input keyword on search input in Question class and send keyword to Server class. The server will call Query method in Searcher class. Next, it retrieve questions that have inputing keyword in InvertedIndex_DB and prioritize them to rank from the most related to the least related information. Finally return questions to the user.



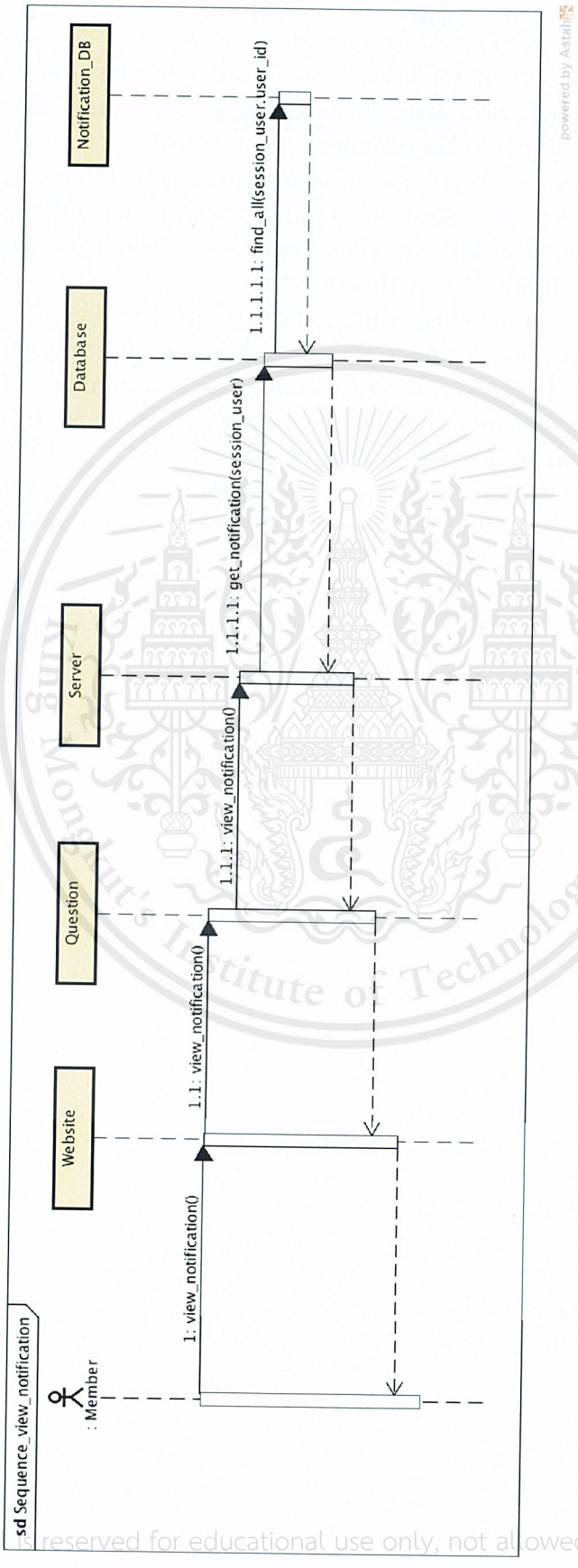
View Member Details

In Figure 4.12 shows a sequence in order to view a member detail. To do that, user will be at MemberProfile class and send a request to get member detail from Server class. The server will get member detail from Member_DB, get questions that created by the member from Question_DB, get answers question that is answered by the member from Question_DB, get vote question that voted by the member from Question_DB, get question that bookmarked by the member from Question_DB. Afterthat, the server send back the data to the Member_DB class.



View Notification

In Figure 4.13 shows a sequence to see all notification user get. To do that, user will press a notification button on header of page, and all notifications will show up. If the user click on one of them, the user will move to page question detail that notification happen



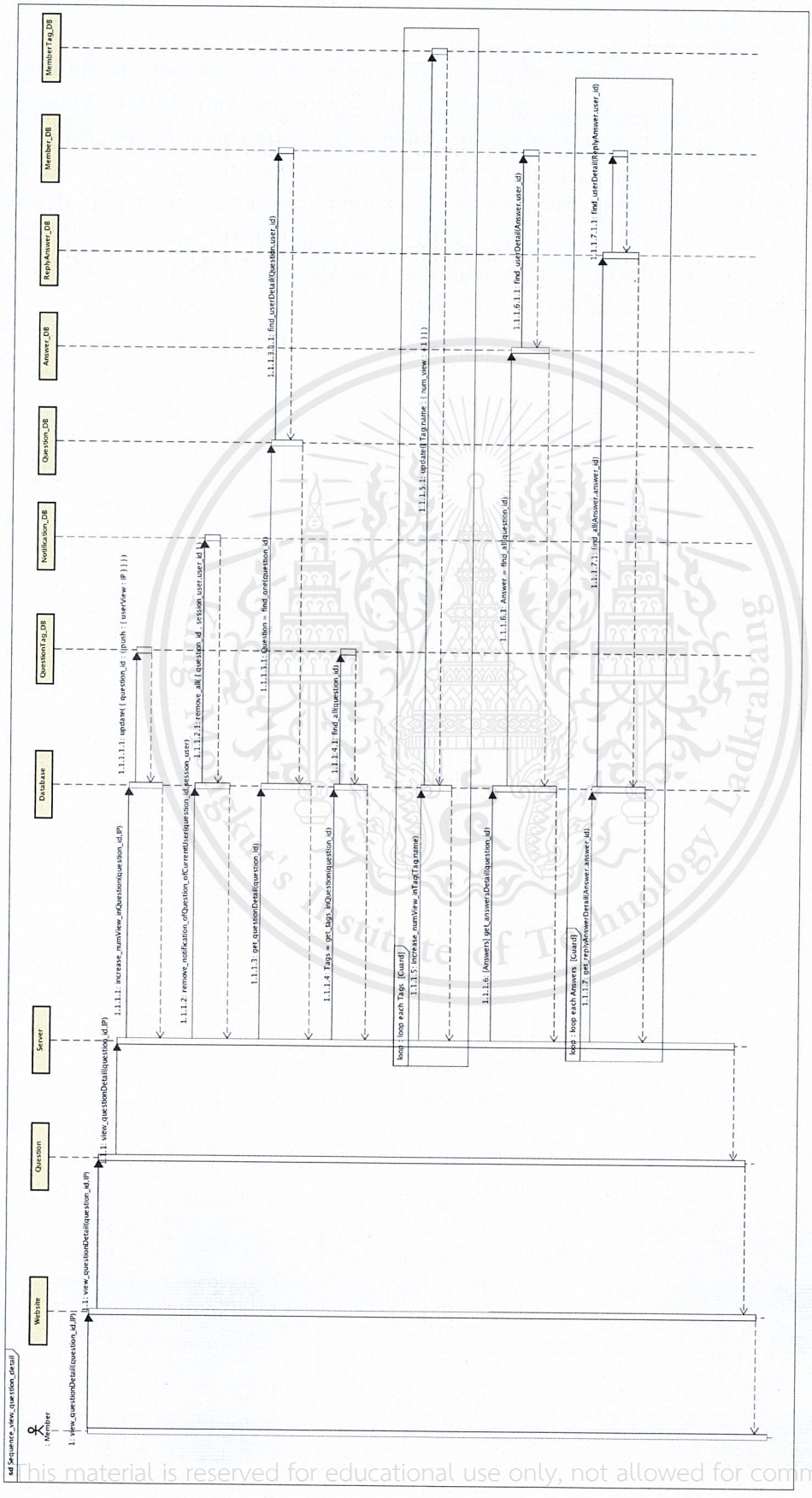
View Detail Question

In Figure 4.14 shows a sequence in order to view question detail. To view question detail, user will be at Question class and send a request to get question detail from Server class.

Firstly, server class will get information to return back by following process. The server will get question detail from Question_DB, get tags of question from QuestionTag_DB, get all answers of question from Answer_DB, get reply answers of all answers in this question.

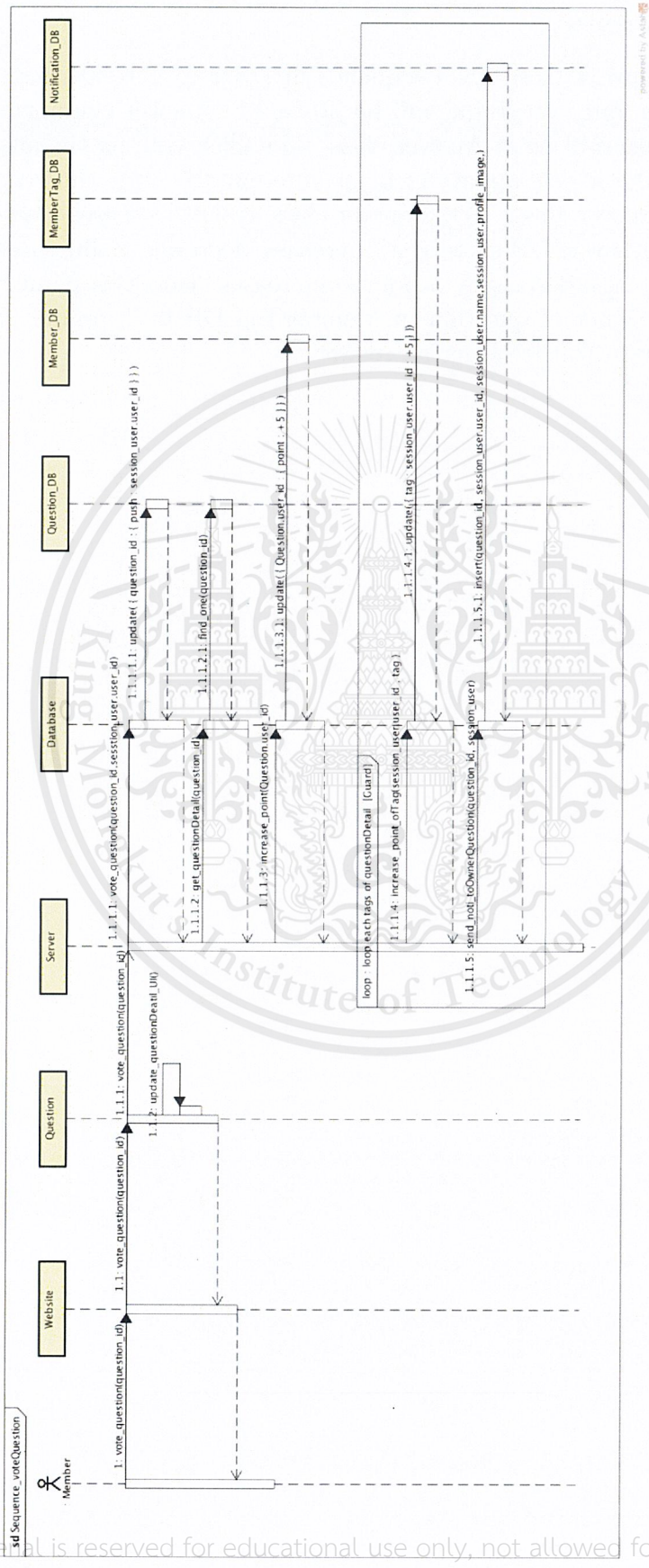
Secondly, server class will update result from viewing this question by following process. Number of view of viewing question will be increased by 1 in Question_DB. Next, it will increase increase number of view of questions' tags by 1 in MemberTag_DB to prioritize the most related questions and suggest them to user for looking it up.

Lastly, server class will remove notification of this current user and current blog in Notification_DB, meaning that the user already visit page where notification come from.



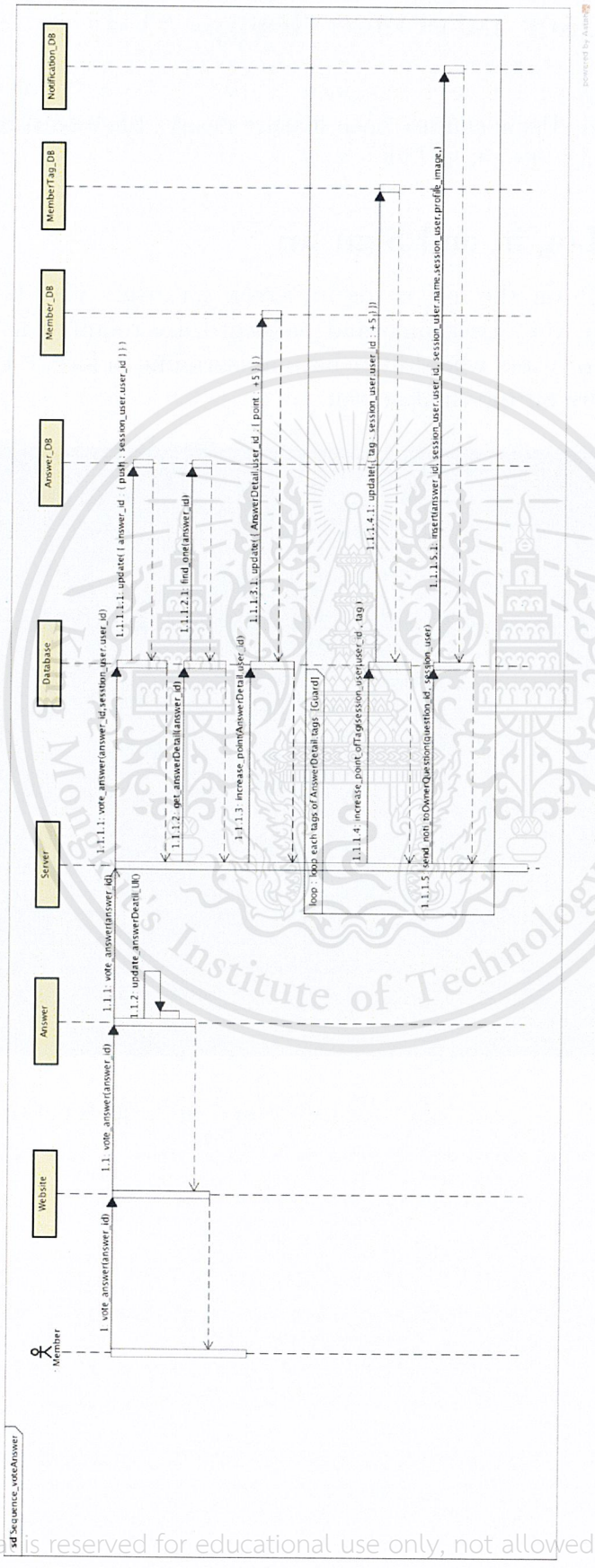
Vote Question

In Figure 4.15 shows a sequence in order to vote question. To do that, user will be at Question class, then click vote on the question detail. The question detail will update to be as voted at the same time the vote request is sent to Server class. In the Server class, it will increases/decreases number of vote in Question_DB class by 1, increases/decreases point to owner of question in Member_DB class by 5 points, increases/decreases point of each question tags to owner of question in MemberTag_DB by 5 points. Lastly it send a notification to the owner of question.



Vote Answer

In Figure 4.16 shows a sequence in order to vote answer. Similar to vote question, but everything will be process in Answer class instead of Question class. user will be at Answer class, then click vote on the answer detail. The answer detail will update to be as voted at the same time the vote request is sent to Server class. In the Server class, it will increases/decreases number of vote in Answer_DB class by 1, increases/decreases point to owner of question in Member_DB class by 5 points, increases/decreases point of each question tags to owner of question in MemberTag_DB by 5 points. Lastly it send a notification to the owner of answer.



4.5 User interface design (UI)

In designing the user interface, it has to be easy for users to use and understand. Users can use each feature easily. Each feature is provided with user interface design as follows:

4.5.1 Log in and Sign up

Figure 4.21 on the left is log-in screen for users who have not logged in the website yet. Username and password are required for accessing to the website. For users who do not have a username before, they need to sign up which shows on the right screen.

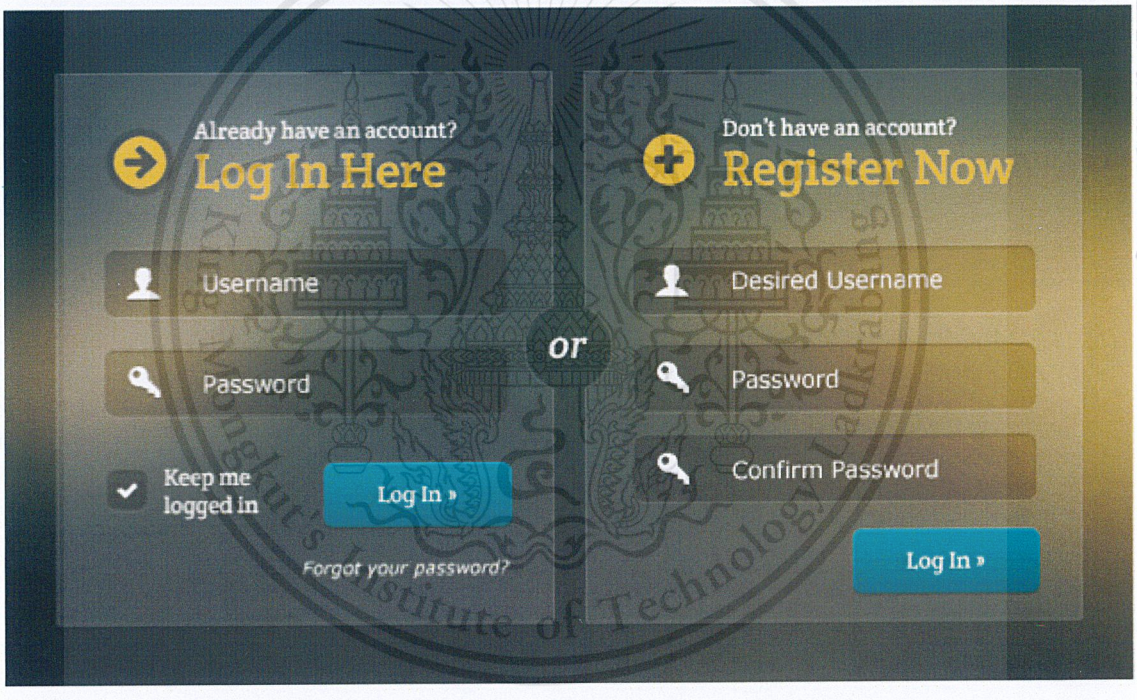


Figure 4.21: login interface and register interface

From figure 4.22, it displays five applications for login or register in Bugcoli. Facebook Twitter GooglePlus LinkedIn and Github respectively

Figure 4.22: login via social network

4.5.2 Non-Login Homepage

Figure 4.23 is non log-in screen for users who have not logged in Bugcoli yet. Important features will be displayed here to introduce the website to the visitors. Users can select each category of questions by picking up the mode at top Figure 4.27 (blog and question can be used interchangeably)

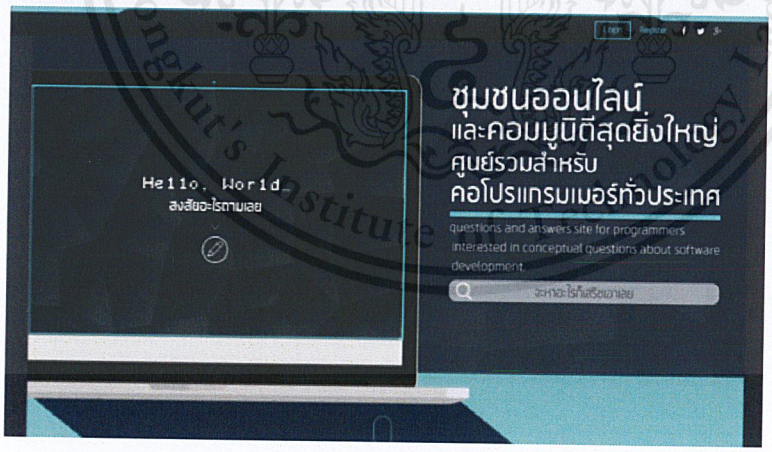


Figure 4.23: Bugcoli homepage for non-login user

Figure 4.24 shows questions in Bugcoli are classified into four categories namely new questions, most vote questions, most views questions, and no answer questions. In addition, questions of each catagory can be sorted out by day, week, month, and forever. Circle with plus symbol is to create new question

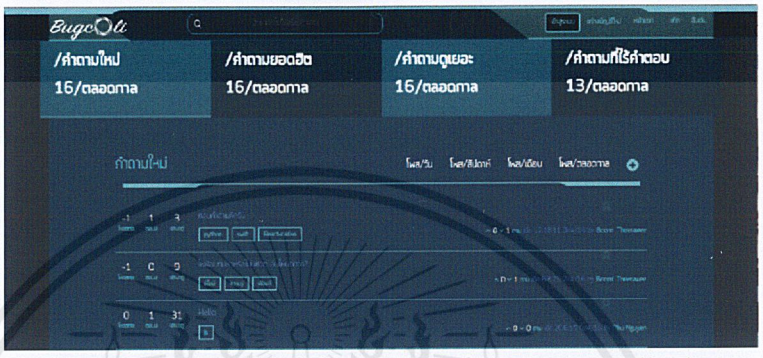


Figure 4.24: Bugcoli homepage body

4.5.3 Banner's content

Bugcoli banner is form of advertising on the homepage. This form of advertising will be slided on homepage that is easy to visible. It is intended to attract visitors to see our features and details on our website. Moreover, considering as a user experience, a banner can be linked to a shortcut feature as well. The following figures show an example of our banners content.

Figure 4.25 shows 4 Banners describing significant feature for Bugcoli. The banner with pencil will be linked to create question and the level ranking banner will be linked to ranking users.

Figure 4.25: Banner slide in advertisement homepage

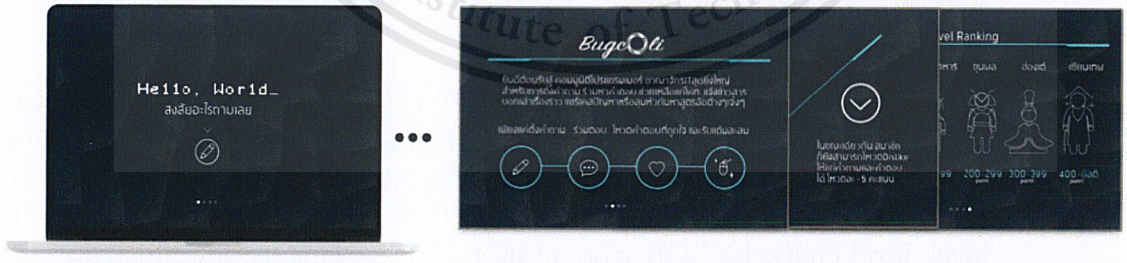


Figure 4.26: Banners are slided from right side to the left side within Macbook screen

4.5.4 Login Homepage

Figure 4.27 shows suggestive questions' screen for users who have already logged in. On this page, list of questions that the user might interest will be shown here. The propose of this feature is to offer unanswered questions to the logged-in users who tend to give an answer for their interesting tags

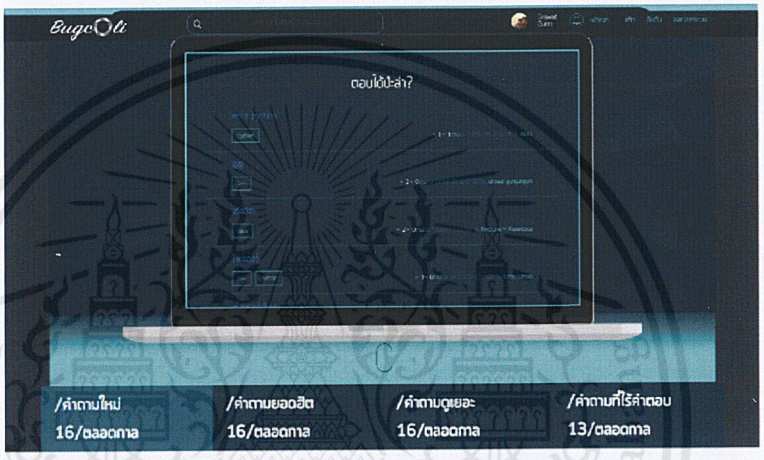


Figure 4.27: Suggest questions

4.5.5 Profile page

In figure 4.28, clicking on the username or the avatar of a user in any pages on Bugcoli website is a way to view the user profile. There are basic information about a user and collection of posts in this page which is divided into two sections. The first section shows user detail in general displaying name, email, level, level-character, and score of user. The second section illustrates statistics of user's professional on the circle right hand side. The second section displays collections of user behavior which contain user created questions, answered questions, voted questions, and bookmarked questions.

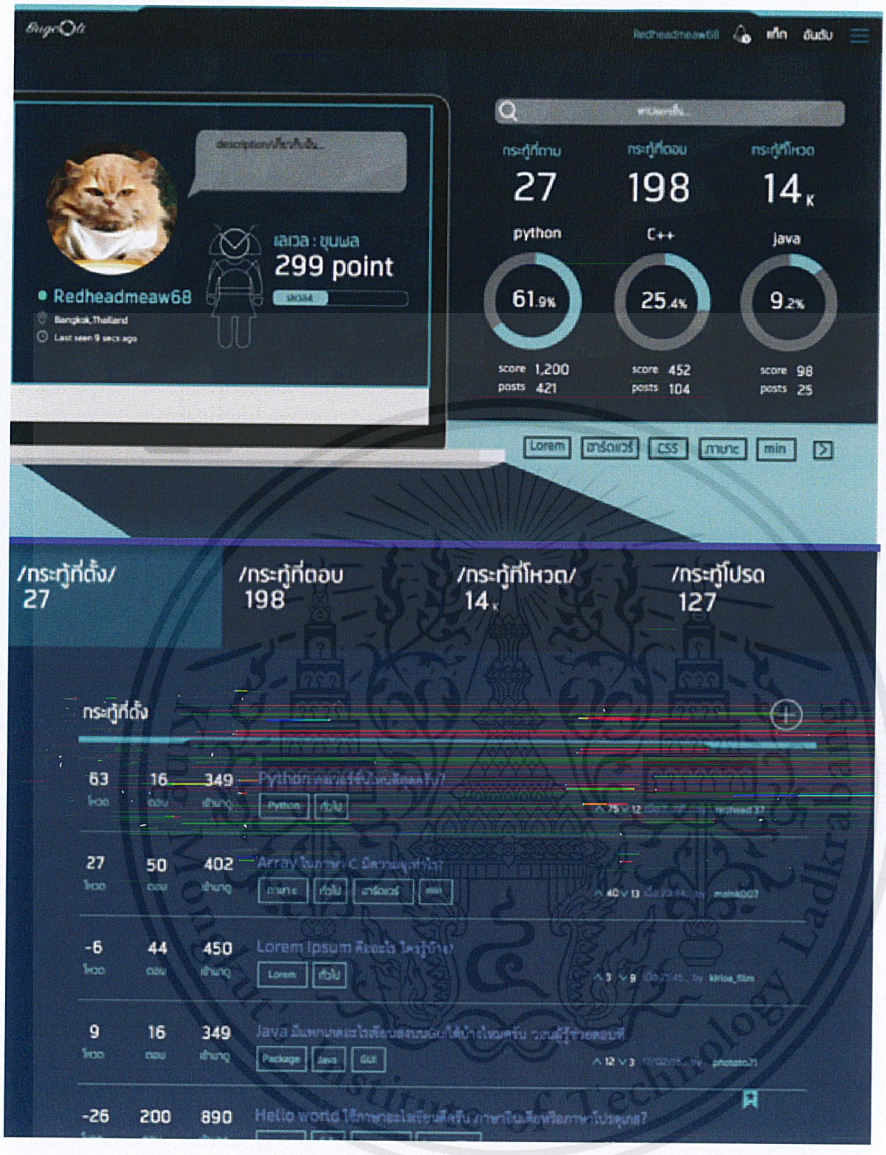


Figure 4.28: User profile page

4.5.6 Tag page

In figure 4.29, all tags in Bugcoli will be displayed as a table on this page. names of tags are cover with border followed up by the number of blog.

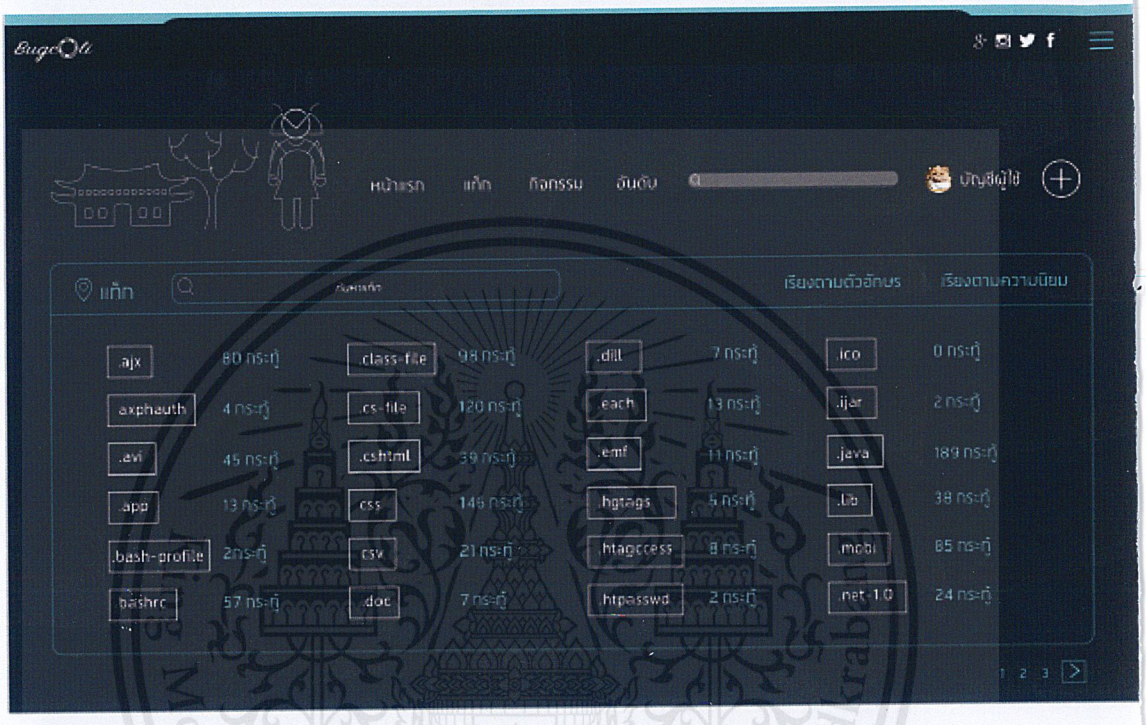


Figure 4.29: Questions tag page

4.5.7 Questions detail page

In figure 4.30, it shows the question contents when user is in question detail page. It shows user detail of question's owner on the bottom right as well as upvote and devote feature. On the top right, the mark shows whether the question has already answered or not. On the top left, it is a bookmarks feature for user to get notification whenever there is a update on question.

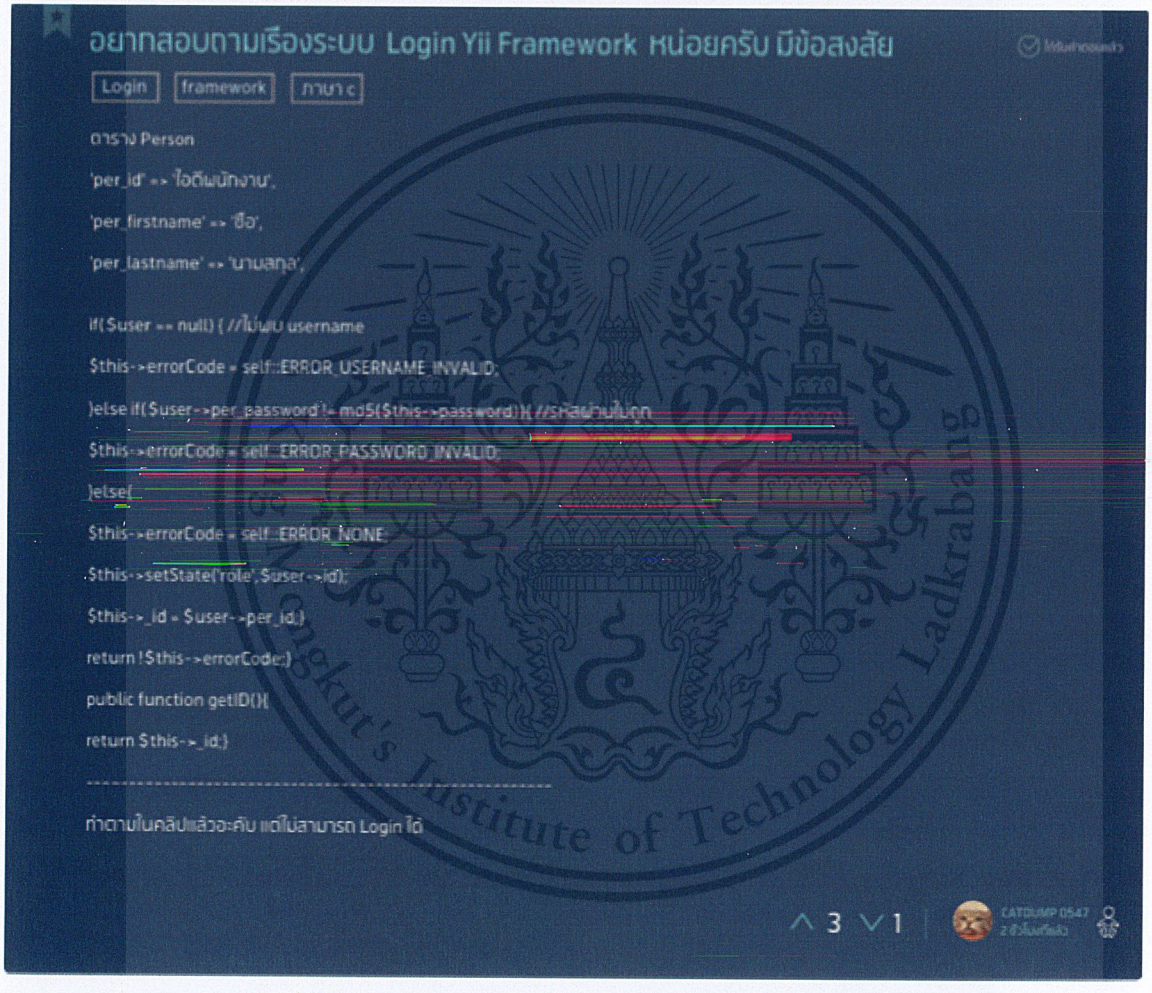


Figure 4.30: Questions content

In figure 4.31, it shows list of answers content. owner of question can choose answer to be the best answer, then that answer will move to the top of question list.

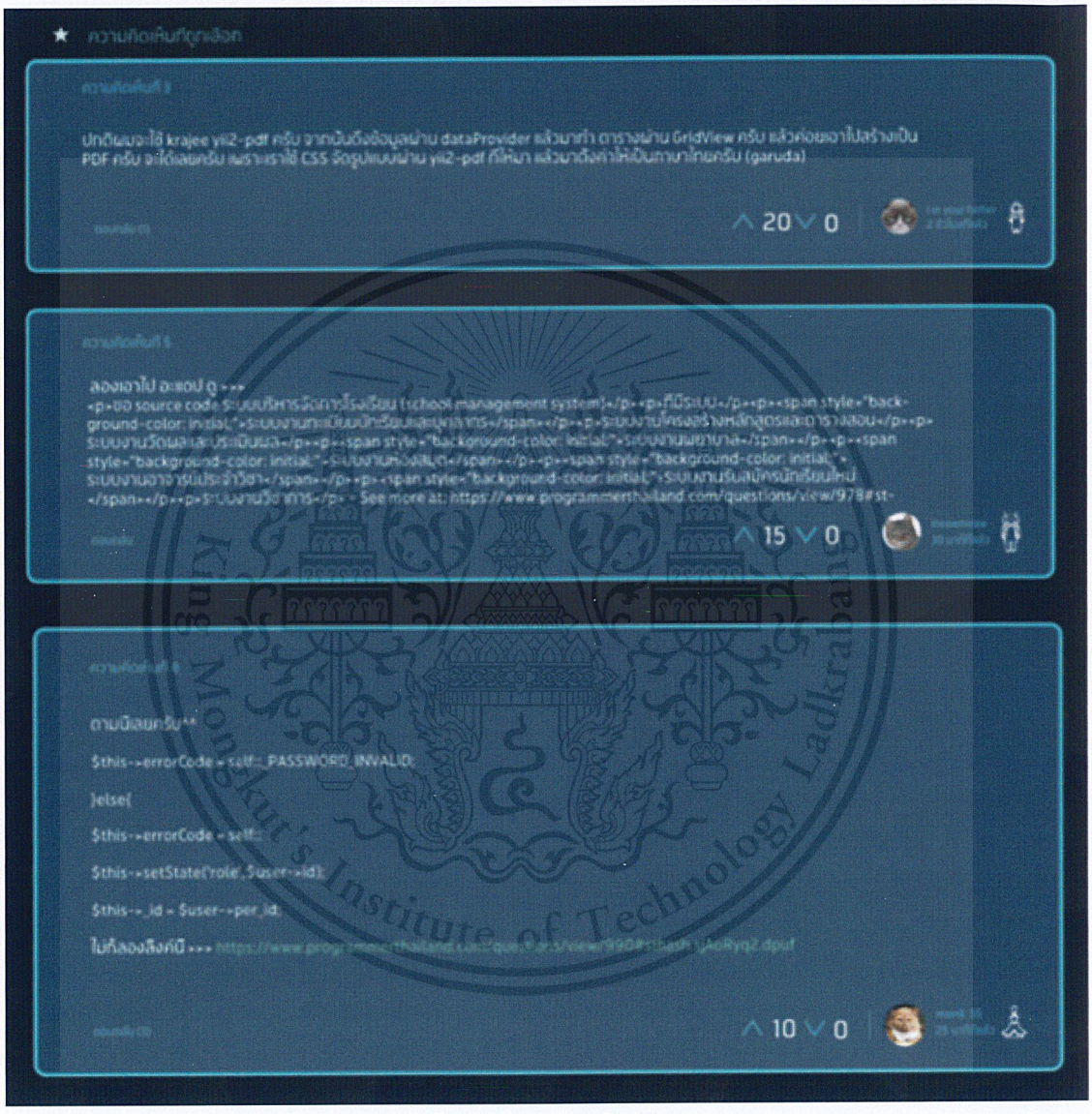


Figure 4.31: Questions answer

In figure 4.32, it shows how UI displays when someone reply in a comment and how to select an answer to be one of the best answers by clicking in the circle.

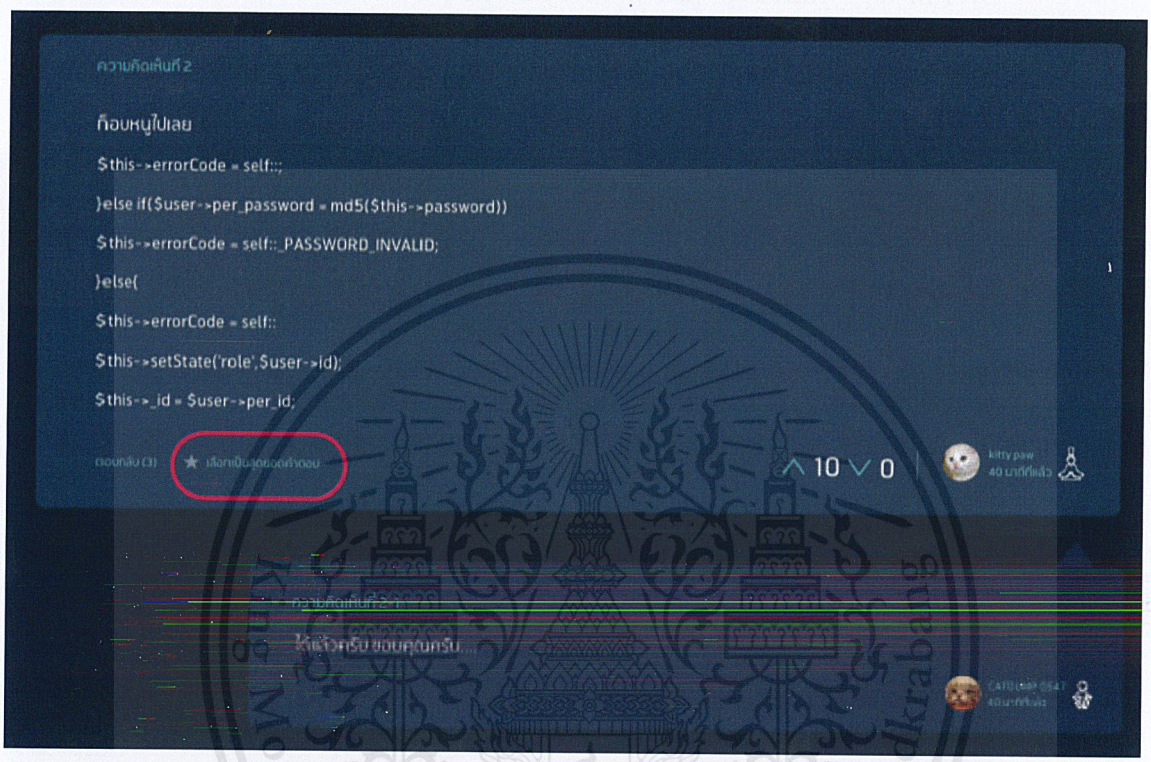


Figure 4.32: Reply answer and select the best answer

4.5.8 User Ranking

In figure 4.33, Users are allocated points for their professional in tags. The overall user rankings are shown in this page. The ranks are divided into 6 characters which display as precedence. When the user clicks on a character on the left hand side, the list of users will be shown from highest score to the lowest.

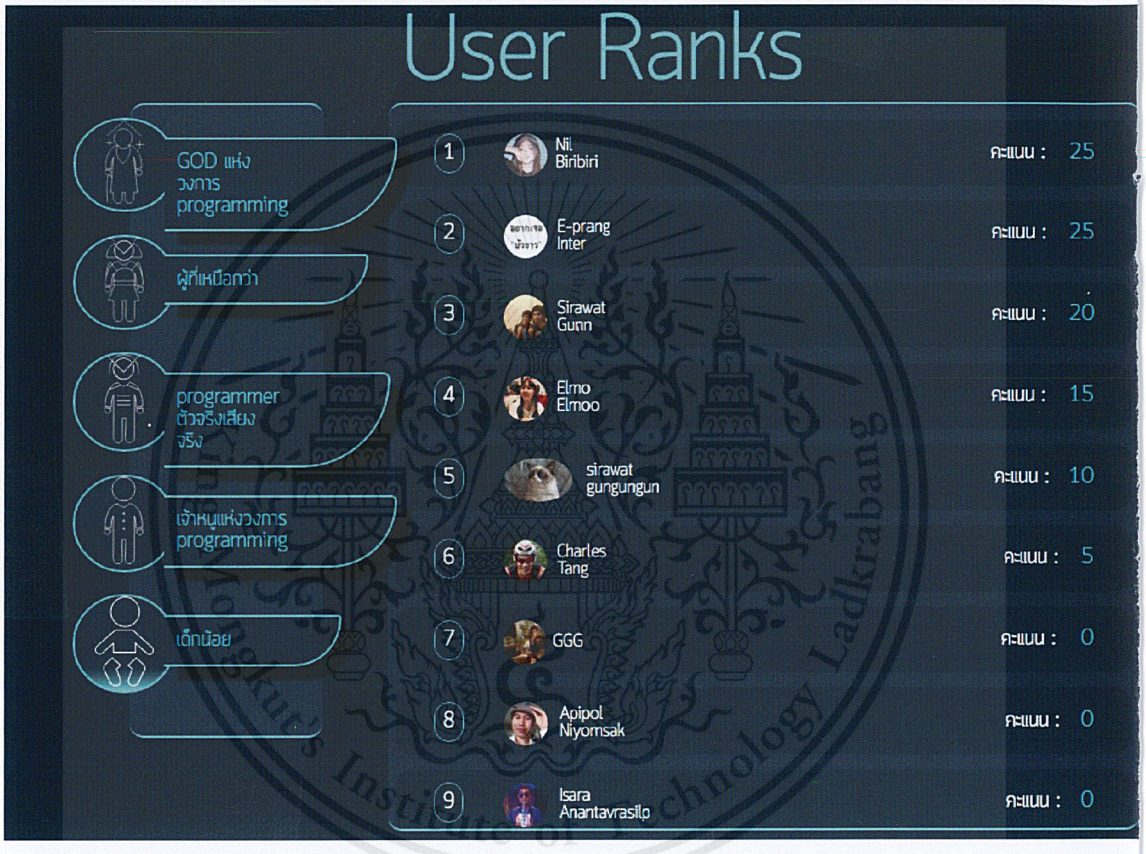


Figure 4.33: User ranking page

4.5.9 Responsive Web Design

Responsive web design is becoming important as the amount of mobile traffic now. Thus, Bugcoli intended to support mobile friendly if the access was made from a mobile device or even a tablet.

Figure 4.34: Bugcoli hompage via mobile device

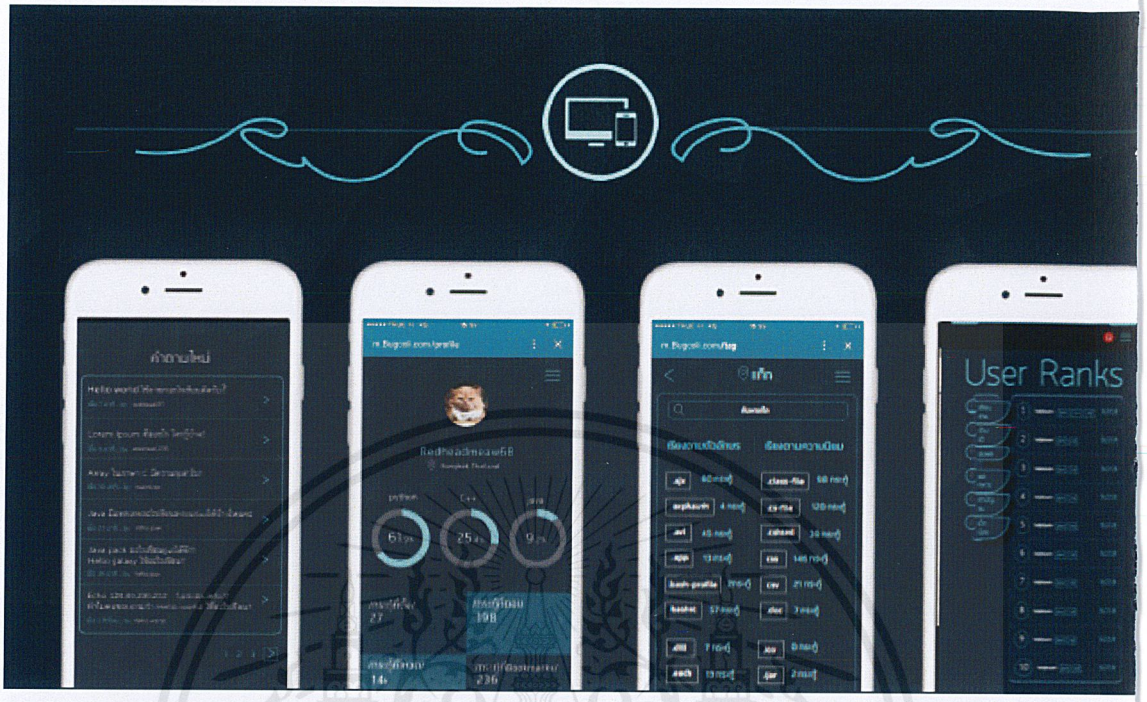


Figure 4.35: Suggest questions, user profile page, tag page, and user ranking page respectively

From figure 4.36, it shows the view question screens including the topic and selected answers [left], comments and each reply on a comment [middle] and a text area to send a comment [right].

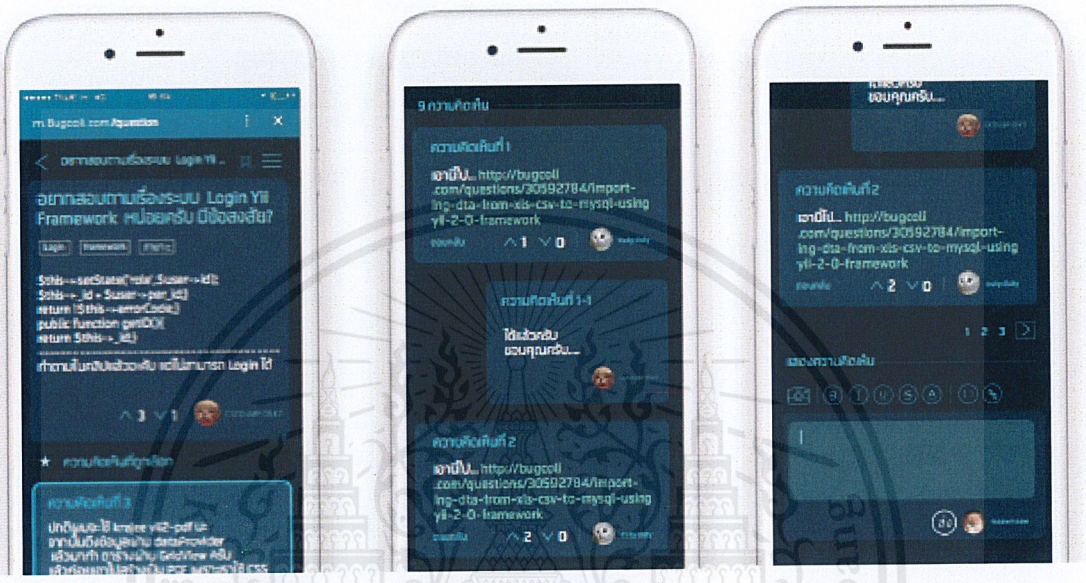


Figure 4.36: Question detail via mobile device

From figure 4.37, When a user presses menu button on the top-right corner, list of pages will be displayed

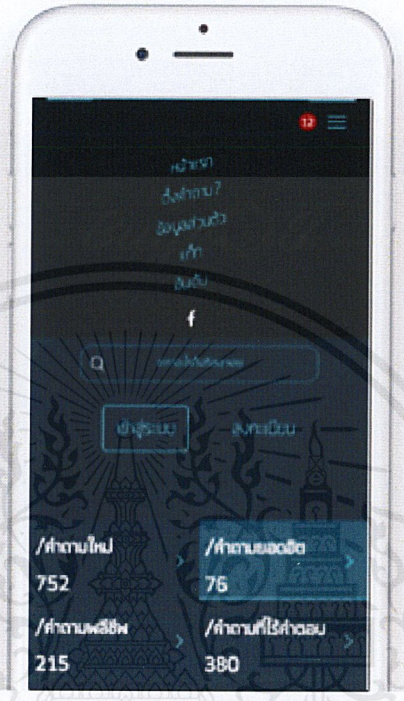


Figure 4.37: Header menu via mobile device

From figure 4.38, notification is slightly appeared on the right when the red circle at top is pressed.

Figure 4.38: Notification pop-up

Chapter 5

Evaluation

An evaluation of the proposed application is given in this chapter. Section 5.1 describes the methodology. Section 5.2 gives results of evaluation.

5.1 Methodology

The project evaluates the proposed application in two aspects: performance, and UI-friendly. We design a two-section evaluation form with respect to these two aspects as follows:

Performance The goal is the measure to efficiency of the application whether the features are good or lack of some performances. There are:

1. Logging in.
2. Creating questions.
3. Answering questions.
4. Replying answers.
5. Accessing question details.
6. Accessing member details.
7. Voting questions/answers.
8. Searching questions.

UI-friendly The goal is to measure the user interface of the application whether the application is understandable or not, whether it is easy-to-use and Convenience to use. There are:

1. Suggesting Questions Section.
2. Users' Rankings Page.
3. Tags Page.
4. Strongest Member Tags.
5. Question Details Page.
6. Sorting Questions in Homepage.
7. Notification Displays.
8. Mobile Device Friendly Display.

These are the rate for the satisfaction level for users:

- Very satisfied = 5 points
- Somewhat satisfied = 4 points
- Neither satisfied nor satisfied = 3 points
- Somewhat dissatisfied = 2 points
- Very dissatisfied = 1 point

5.2 Results

Performance	Efficient		
	Average	Standard Deviation	Quality
1. Performance			
1.1 Logging in	5	0	Excellent
1.2 Creating Questions	3.2	0.83	Good
1.3 Answering Questions	4	0.7	Very good
1.4 Replying Answers	4.2	0.83	Very good
1.5 Accessing Question Details	4.8	0.44	Very good
1.6 Accessing Member Detials	5	0	Excellent
1.7 Voting Quesitons/Answers	4.4	0.54	Very good
1.8 Searching Questions	4.2	0.83	Very good
The Overall Average	4.35	0.52125	Very good
2. UI-Friendly			
2.1 Suggesting Questions Section	4.4	0.54	Very good
2.2 Users' Rankings Page	4.4	0.54	Very good
2.3 Tags Page	4	0.7	Very good
2.4 Piechart of strongest Member Tags	4.2	0.83	Very good
2.5 Question Details Page	4.6	0.54	Very good
2.6 Sorting Questions in different catagories in Homepage	4.2	0.83	Very good
2.7 Notification Displaying	4.6	0.54	Very good
2.8 Mobile Device Friendly Display	4.4	0.54	Very good
The Overall Average	4.35	0.6325	Very good

Table 5.1: Overall Average

Chapter 6

Conclusion and Suggestion

6.1 Conclusion

The aim of this project is to create a promising community for Thai programmers to gather together. Thai programmers here discuss or exchange ideas and help others on their difficulties in programming field. Bugcoli is fully committed to serve Thai programming community with a webboard, which programmers in Thai society can ask questions and answer others. The community also has its own freedom to decide reliability on contents in the website by giving votes. Bugcoli also gathers information of each member to provide a better perspective and reliability of themselves for the community. Users can register and authenticate via a personal account or various choices of social media. Many questions and various categories will classify questions with tags. Helpful questions to certain members, but not popular to the community may be difficult to browse to. However, with the bookmark service, members can mark their desired question and conveniently access it anytime by browsing through their own bookmarks list. In addition, all activities related to the member, happening all the time, in the community is always reported by the notification functionality. The website user interface design is responsive, allowing users to be able to access the system from various devices and the Internet web browsers. The designs and graphics of web-page components are in a modern style and up-to-date fashion in 2016. After an evaluation, the results are quite satisfiable. The website is on advertisements in social medias, reaching out to numerous Thais. On Facebook advertisements, the web site's self-introduction advertisement have

reached to at least 1,000 viewers, and is expected to reach out more. There has currently been no complaining feedbacks by emails. In fact, no emails for the website have been received at all, despite that the website is expected to allow users to email comments demanding for improvements and wishes towards developing future functionalities.

6.2 Suggestion

Bugcoli is expected to expand its community to serve an optional solution to Thai programmers with programming difficulties. Despite existing extremely big international websites with a base of at least 1 million members, many Thai programmers are still being barricaded by the language barrier. International websites use English as their main language for communication. Most programming languages, especially all the popular programming languages are English based. However, Thai programmers still tend to understand just only how each keyword in each programming language works, but does not understand and have poor interpretation with real English communications according to average statistics of many standardized English tests. Bugcoli must adapt itself in the world of advertising and introduce itself to more Thai programmers who are still in the international community. This will depend on the marketing strategies of the website, which has never been conducted.

Bibliography

[1] Sitanath Biswas Ajit Kumar Mahapatra. Inverted indexes: Types and techniques. *IJCSI International Journal of Computer Science Issues*, 8(1), jul 2011.

[2] Jeff Atwood. What was stack overflow built with?, may 2016.

[3] Phil Bagwell. Fast and space efficient trie searches. Technical report, Ecole polytechnique federale de Lausanne EPFL, Es Grands Champs, 1195-Dully, Switzerland, 2000.

[4] BuiltWith. Pantip.com technology profiler on builtwith, may 2016.

[5] Alisa Kongthon Choochart Haruechaiyasak. Lextoplus: A thai lexeme tokenization and normalization tool. In *The 4th Workshop on South and Southeast Asian NLP (WSSANLP), International Joint Conference on Natural Language Processing*, 2013.

[6] Hinrich Schütze Christopher D. Manning, Prabhakar Raghavan. *Introduction to Information Retrieval*. Cambridge University Press, jul 2008.

[7] Andri Mirzal. Design and implementation of a simple web search engine. *International Journal of Multimedia and Ubiquitous Engineering*, 7(1), sep 2001.

[8] Juan Ramos. Using tf-idf to determine word relevance in document queries. Technical report, Department of Computer Science, Rutgers University, 23515 BPO Way, Piscataway, NJ 08855, 2003.

[9] Apisilp Trunganont. Why pantip is often closed for maintenance? lessons, pain, and tears from mongodb - macroart, oct 2013.

Chapter 7

Appendix A

No.	Project steps	Who	21/9/2015	5/10/2015	19/10/2015	2/11/2015	16/11/2015	7/12/2015	21/12/2015	11/1/2016	25/1/2016	8/2/2016	22/2/2016	7/3/2016	21/3/2016	18/4/2016	2/5/2016	9/5/2016
1	**Identify and define system features**																	
	list all features	All																
	gathering requirements	All																
2	**Study**																	
	study Go	Gun																
	study ReactJS	Gun																
	study MongoDB	Gun																
	study Search algorithm	Tong																
	study Gulp	Boom																
	study Jade/Sass/Es6	Boom																
3	**Design components of the system**																	
	sitemap	Boom																
	design webpage	Boom																
	homepage	Ming																
	profile page	Ming																
	tag page	Ming																
	rank page	Ming																
	create blog page	Ming																
	view blog page	Ming																
	Responsive for mobile devies	Boom																
	design database	Gun																
4	**Implement**																	
	search tools preparation	Tong																
	implement search	Tong																
	integrate with website	Tong																
	create blog	Gun																
	comment blog	Gun																
	upvote, devote blog,commnet	Gun																
	reply comment	Gun																
	rich text editor	Gun																
	store image, show image to url	Gun																
	integrate reactJs with interface(html/css) that has finished	Gun																
	sort blog	Gun																
	store IP of user when seeing blog in order to know how many view of this blog	Gun																
	notification	Gun																
	update score after click upvote, devote, and when select best answer	Gun																
	Coding webpage	Boom																
	homepage	Boom																
	profile page	Boom																
	tag page	Boom																
	rank page	Boom																
	create blog page	Boom																
	view blog page	Boom																
	convert Jade to ReactJS	Gun																
	Responsive website	Boom																
5	**Test website and fix bugs**																	
	Check links	Boom																
	UI/UX Testing	All																
	Beta release	All																
	test search feature	All																
	test website and fix bugs	All																
	implement remained functions	All																
	release completed website	All																
6	**work on document**																	
	documentation	ALL																