Pre-Final Report

Mongbon

Propose to

Assist. Prof. Dr. Thanarat Chalidabhongse

Kasidit	Iamthong	5731005321
Kawin	Liaowongphuthorn	5731004721
Kosate	Limpongsa	5731012721
Parinthorn	Saithong	5730329521
Sirinthra	Chantharaj	5730635521
Thanat	Jatuphattharachat	5730243121

This project is a part of a course 2110332 Systems Analysis and Design

Second Semester, Academic Year 2016

Computer Department, Faculty of Engineering, Chulalongkorn University

Table of Contents

List of Figures

List of Tables

1. Windows Navigation Diagram

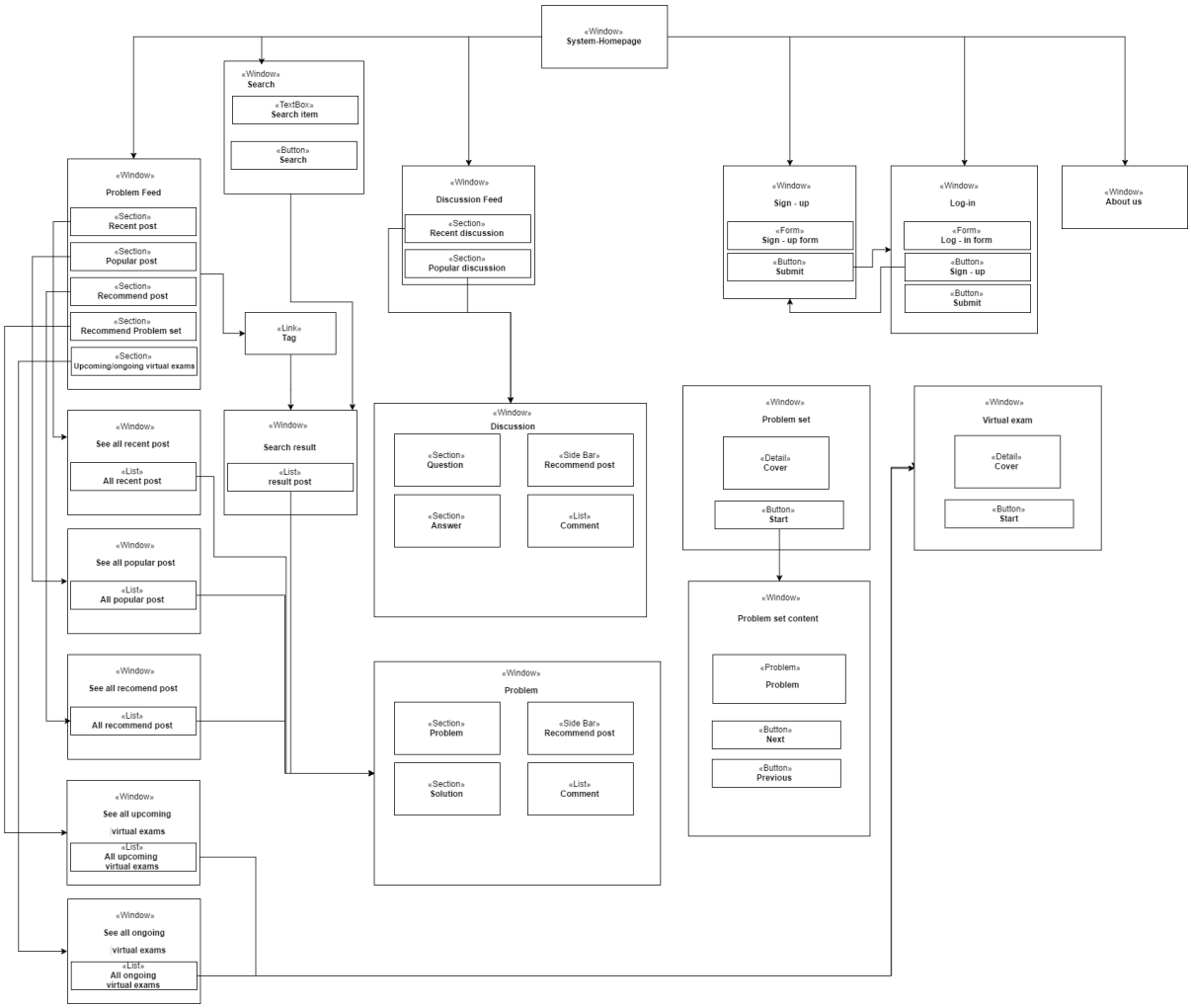

Figure 1-1: Windows navigation diagram for non-member

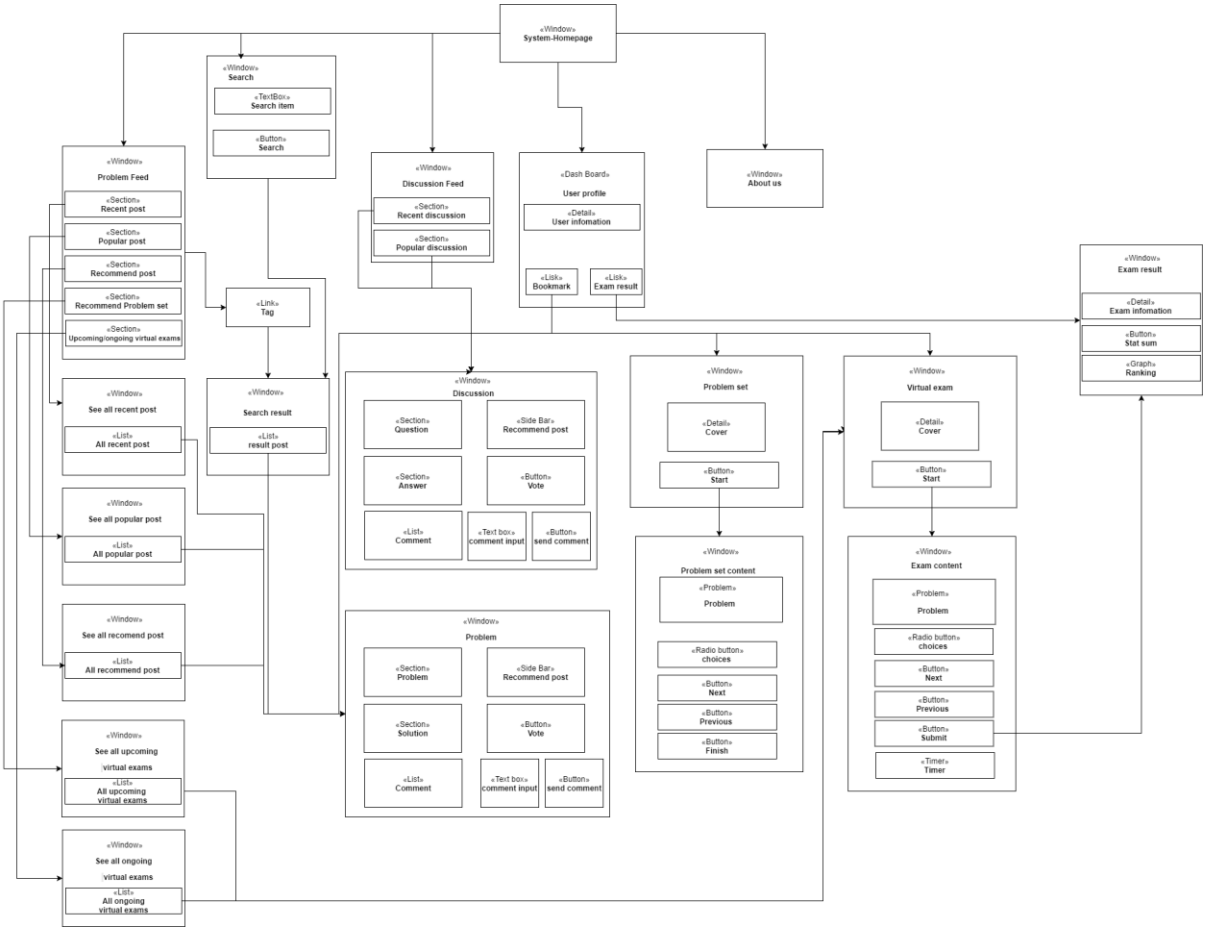

Figure 1-2: Windows navigation diagram for member

2. User Interface Design Principles and Techniques

To deliver ease of use to system users, several design principles are considered during the user interface design process. The principles used in such process are listed below:

2.1. Layout

Each page consists of 3 prominent sections as shown in Figure 2-1: navigation bar at the top of the page, main content in the center, and footer at the bottom. First, the navigation bar section provides entrances to main features along with primary actions so that users can readily navigate and browse through an application. Next, the main content section serves data to users based on the page purpose. Last, the footer section categorizes links as a minimal sitemap to facilitate users finding pages as they need. In addition to the main sections described previously, a sidebar on the right next to main content section can be included to provide secondary data about associated content on that page. Finally, the main sections can be subdivided and manipulated but the basic structure should remain relatively consistent.

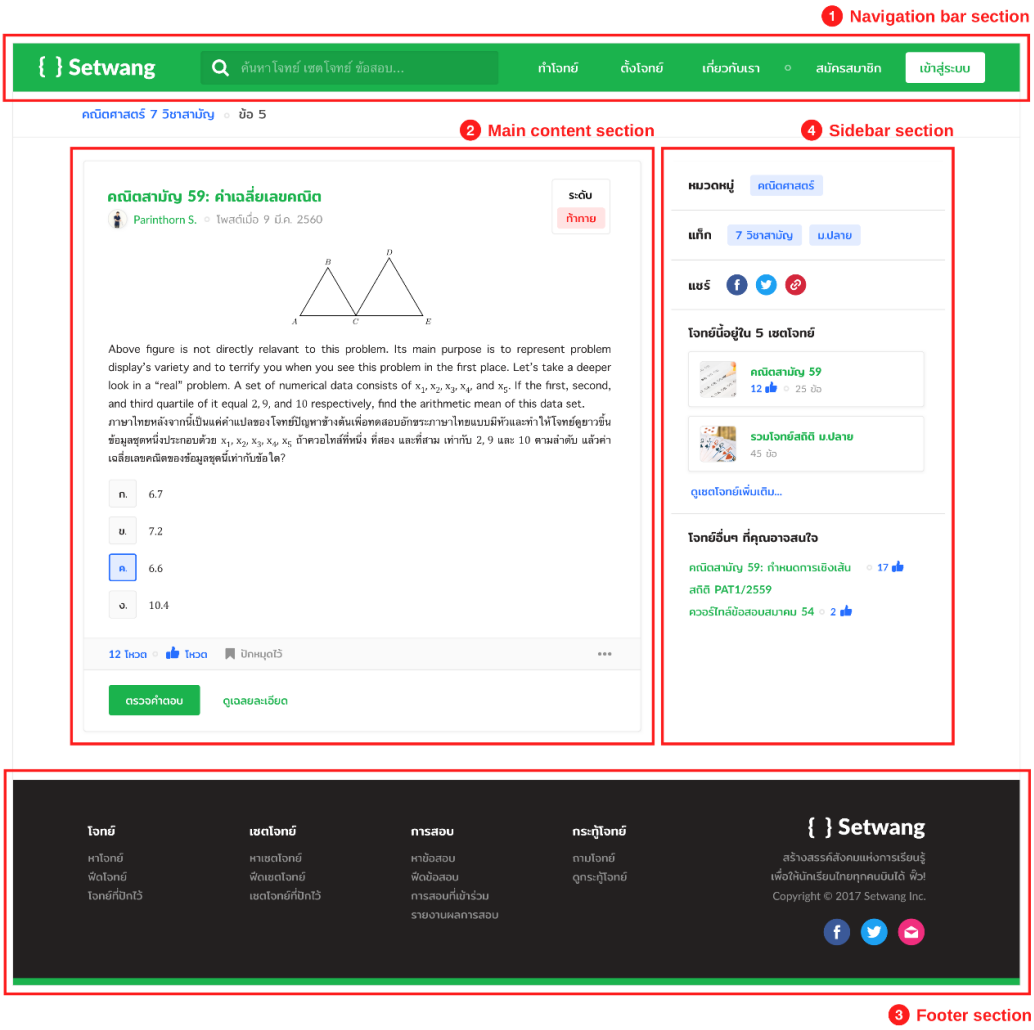

Figure 2-1: Layout

2.2. Content Awareness

Page title and breadcrumb should be included in all main pages, as shown in Figure 2-2, to indicate where the users are. This also aids them easily going back and forth, since breadcrumb keeps track of their location within the website. To be clear, all interfaces should have titles. Field labels should be short and specific; explanations should be provided if necessary. Moreover, raw data and actions should be processed and divided into semantic appropriate sections so that the users will be able to access and utilize the content within the interface effectively.

Figure 2-2: Content awareness aids

2.3. Aesthetics

The aesthetics of the interfaces refers to the structure and designs ability to please the users eye. The following are the guideline in brief. All text should be about the same font size except that we want to distinguish between different types of information. Moreover, a combination of white background and black serif font type should be applied to content in order that it is clearly legible. Figure 2-3 below represents the visual style guide defined to accelerate design process while taking aesthetic into account.

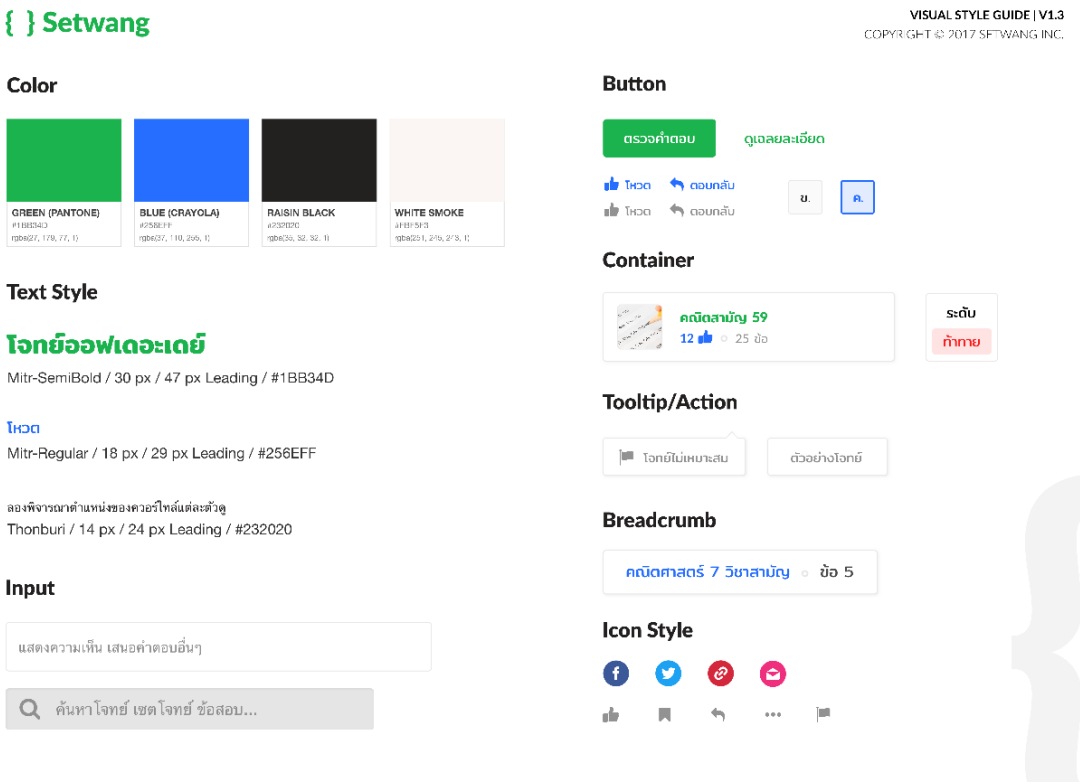

Figure 2-3: Visual style guide

2.4. User Experience

In order to provide ease of learning for novice users and ease of use for experienced users, the user interface should be balanced between quick access to commonly used features and guidance through less well-known ones. Before designing, customer journey and user flow should be carried out. Moreover, interactive evaluation should be performed to make sure that the system is easy to learn and utilize as expected.

2.5. Consistency

An application will be easy to use if there is consistency in the design of the interface. Some consistency rules are listed as follows:

2.5.1. Re-use the same elements for different situations, e.g. design a sample notification and color-code it for different situations.

2.5.2. Align everything nicely along the grid, or introduce any other kind of visual order.

2.5.3. Use a consistent color scheme as shown in Figure 2-4 throughout the app.

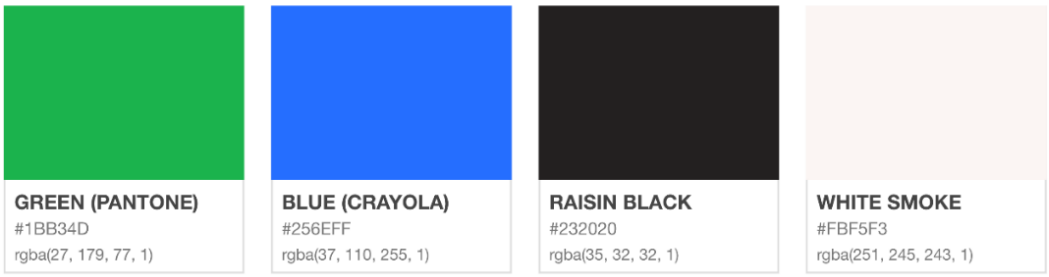

Figure 2-4: Color scheme

2.5.4. Keep the navigation consistent across all screens.

2.6. Minimal User Effort

Components and pages should be logically organized in order to reduce amount of thinking a user need to accomplish a task. The three-clicks rule can be optionally applied.

3. Design Process and User Interface

3.1. Creating Persona

To develop and design user interface from our little understanding of actual users to meet both system requirements and ease of use, we decided to firstly create a persona which can be used to guide our team in unbiasedly mocking up user interface as well as designing customer journey experience throughout the system from a user perspective. This prevents generating the plethora of features and visual components which are unnecessary and can overwhelm a user at the first glance.

Diving into a persona we created, her name is Fah-Sai. She is 17 years old and lives in Bangkok. As in the technology era, she has a smartphone and desktop computer. Since she is still a high school student, her primary dream is to be eligible for the prominent university. In addition, she is always keen on learning new things by herself, so she tends to find learning resources, such as online/offline exercises, courses, books, etc., in order to practice and improve skills in her free time.

As aforementioned, we used Fah-Sai as a persona to shift our perspective to an actual user one. However, a persona can be changed over the time when we find out and realize that current persona is no longer relevant to our actual user. The change of persona can occur for example after initially rolling out the system.

3.2. Sketch and Wireframe

After creating a persona, we sketch the visual interface by converting from textual system requirements in order to consolidate them to be more concrete and to facilitate communicating ideas among our team. We intentionally used marker pens to roughly sketch, since we do not want a design in this stage to be too much elaborated, but we want to keep the interface at high level. Then, all sketches from each team member were reviewed one at a time and were merged to construct a wireframe of each page. In a nutshell, a wireframe is de facto a low-fidelity user interface. We created a wireframe to arrange components to best fit in a screen without concerning the corporate identity. Figure 3-1 below illustrates the problem page's wireframe.

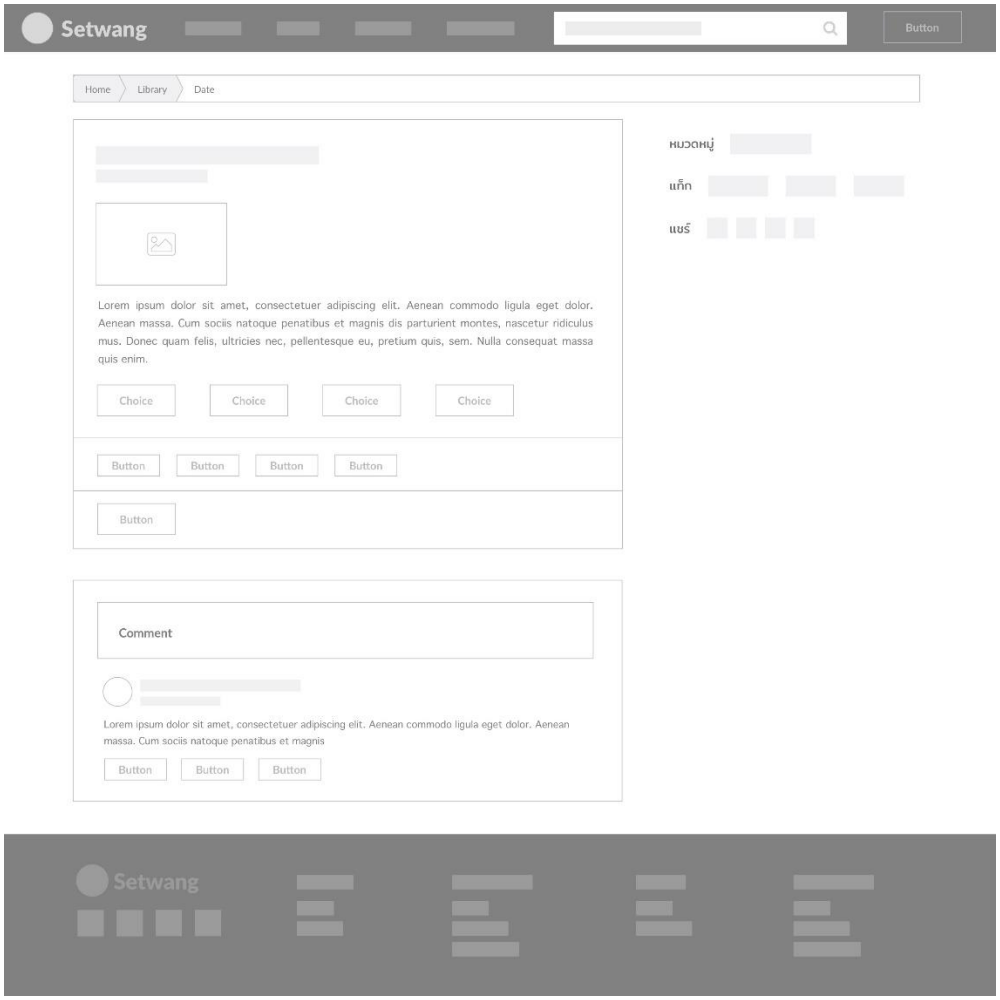

Figure 3-1: Sketch and wireframe

3.3. High-fidelity User Interface

In the next step, each wireframe was streamlined by taking aesthetic into account. A primary product of this step is the high-fidelity user interface which is later fed to front-end development stage. In order to create it, we firstly developed the corporate identity and its byproduct, a visual style guide, which define a guideline for every aesthetic component as shown in Figure 2-3. Their main advantage is to let developers and designers work concurrently while keeping the design congruent and consistent across all pages. After that, the high-fidelity user interface was created by applying such the corporate identity along with the style guide to a wireframe. The Figure 3-2 illustrates the high-fidelity user interface of a problem page deriving from its wireframe as described.

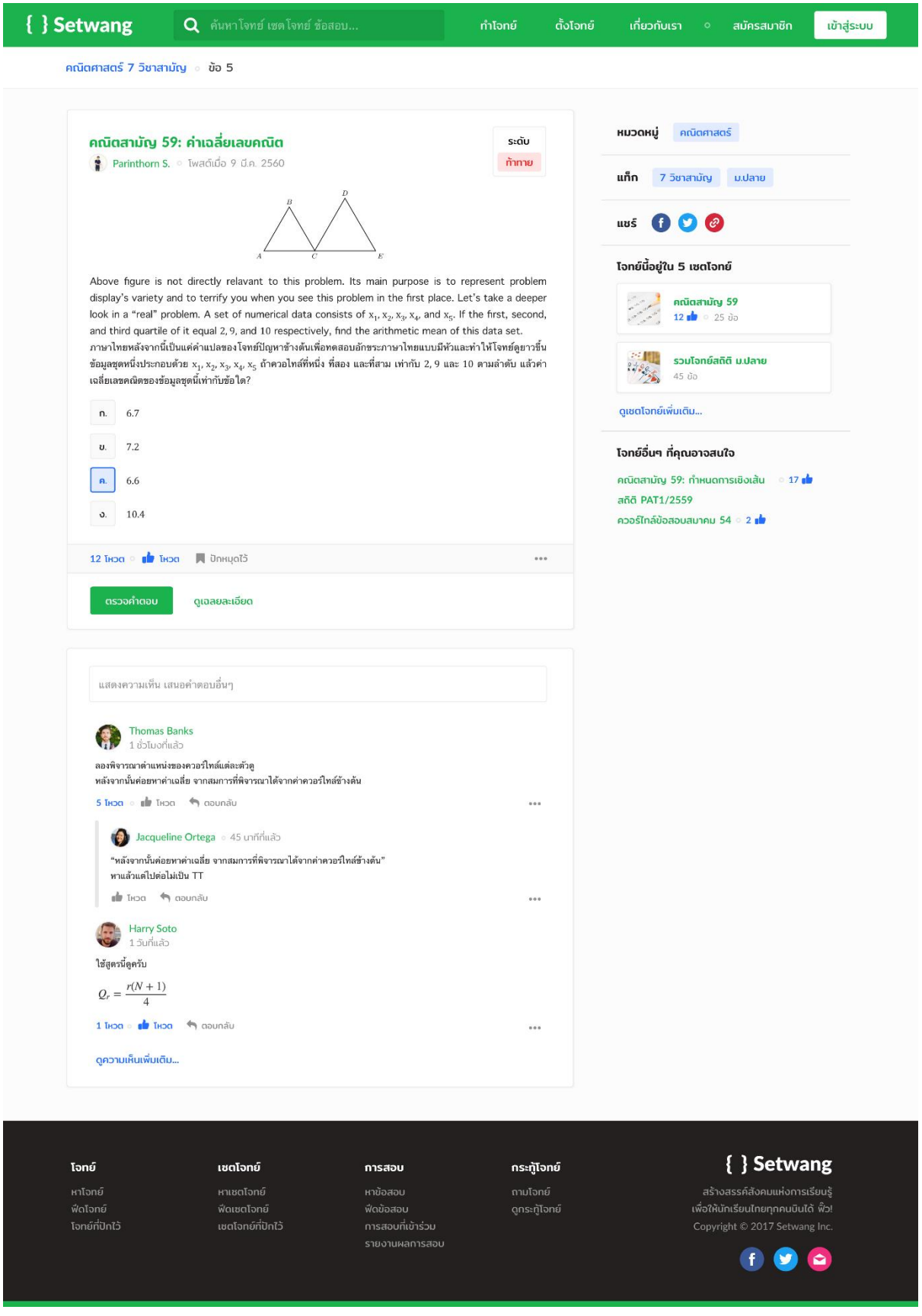

Figure 3-2: Problem page

The breakdowns of this page are discussed in terms of their design background and underlying purpose in the following sections:

Figure 3-3: Navigation bar

As shown in the Figure 3-3 above, the search bar was included right away in a navigation bar. It helps a user browsing through problems, problem sets, and virtual exams in our system. Moreover, we emphasize the login button at the top right corner because we want to encourage a user who come across to the website to engage with the system as a member. Below the navigation bar, we added breadcrumb to aid users easily going back and forth, since it keeps track of their locations within the website.

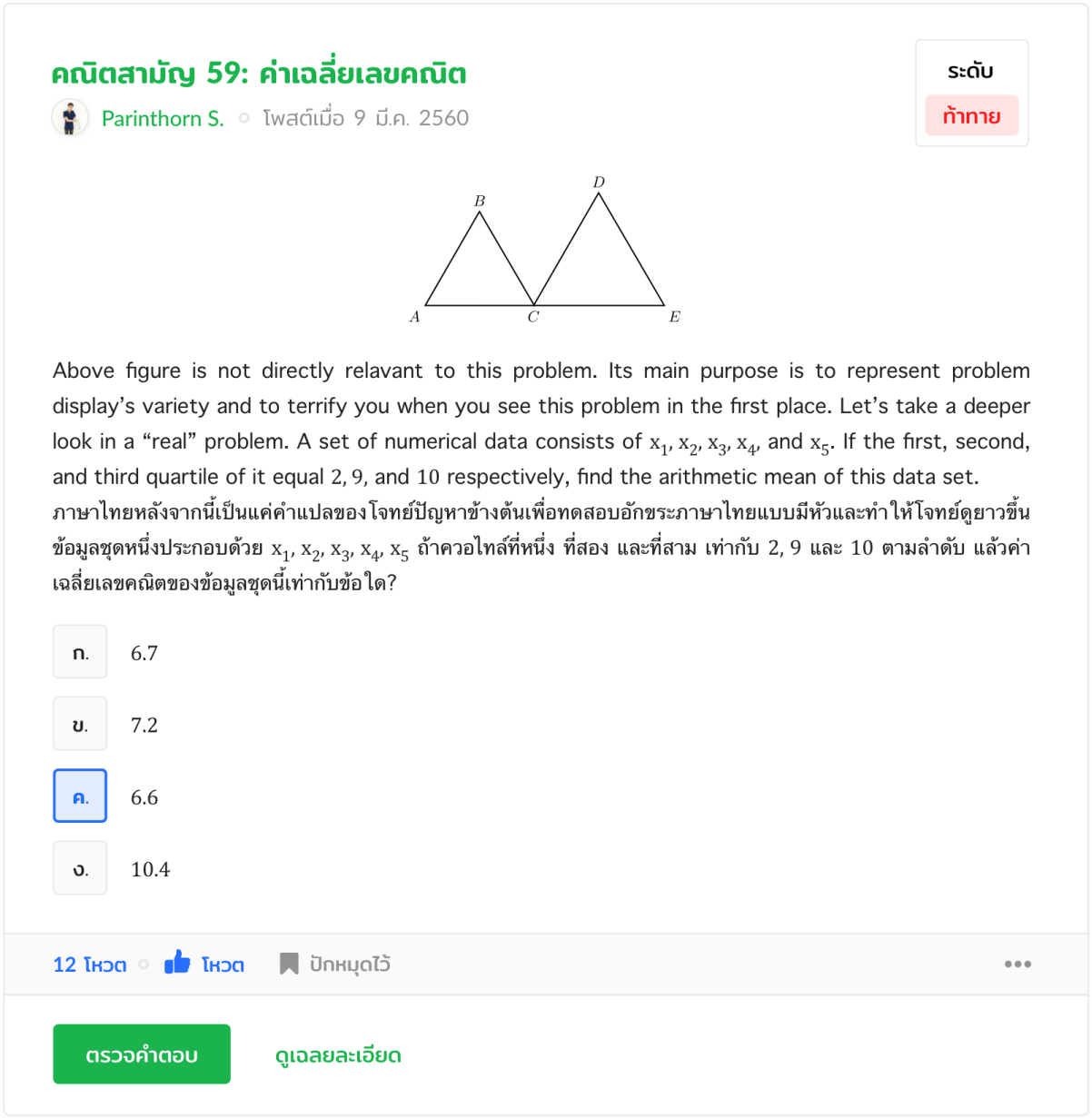

Figure 3-4: Problem box

The next component is a problem box which contains all problem details as shown in Figure 3-4. Because we want it to be compatible with all problem styles, it is designed to be as simple as possible. To achieve that, we decide to use a combination of white background and black serif font type to assure that problem content is clearly legible. We also emphasize the check answer button over the view detailed solution one, because we motivate users to initially solve a problem instead of moving toward to view its solution directly.

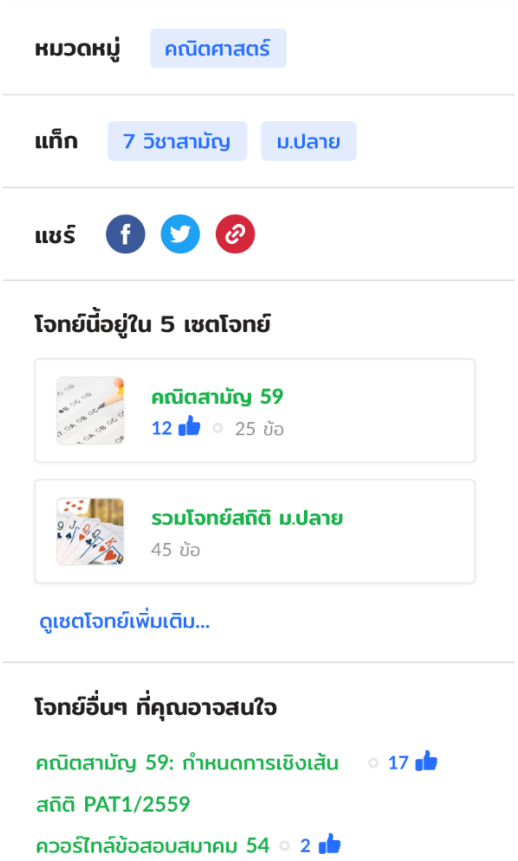

Figure 3-5: Side section showing problem's metadata

As shown in the Figure 3-5 above, a side section provides the various ways to find other problems to solve after having done with the current one. For example, users can view the remaining problems in the same problem set or reach to a problem in recommendation section. Furthermore, users can also share it to the social media including Facebook, Twitter or share by a permanent link. This helps the system gain more audiences from those channels and keeps the community alive.

4. Design System to Support Nonfunctional Requirements

In order to meet nonfunctional requirements, we have to take such requirements into consideration during system design, including class and method design, user interface design, and physical architecture design. Table 4-1 below shows the design solution against each nonfunctional requirement.

Table 4-1: Design solution against each nonfunctional requirement

No.	Type	Nonfunctional requirement	Design solution
1	Operational	The system should operate as a web application in the modern browsers.	Check browser compatibility of dependencies integrated in the system, e.g. frameworks, and libraries, to ensure that the application can serve to users flawlessly in those browsers.
		The system should provide an understandable and easy-to-interact interface for target users.	Follow user interface principles as described in X.X.X and carry out interface evaluation process to ensure that the system is easy to learn and utilize.
		It should not be possible to delete a problem data entirely from the system.	Add a delete flag to database schema to mark a problem as deleted instead of removing it permanently from the system.
2	Performance	The system should handle at least 100 concurrent requests per second.	Design system application to be scalable and deploy it on the suitable hardware.
		The system average response time should be lower than 4 seconds.	Design system application to be scalable and deploy it on the suitable hardware.
3	Security	Only registered user can access to problem's solution.	Design a method to authenticate users and design user interface for logging in and signing up.
		Only system administrators can review flags.	Design a method to authenticate administrators and provide interface for reviewing those flags.
		The private user data must not be accessed without a permission.	Design a method to authenticate users and check user's access right when users request data.
4	Cultural and Political	All problems in the system should not violate the copyright policy.	Include a link to terms and conditions in sign up interface to state application rule and limitation. Provide report mechanism to let users report infringed problems. Log usage and events in the database so that administrators can trace back such violation.
		Copying of other users' problem is not allowed in any case.	Include a link to terms and conditions in sign up interface to state application rule and limitation. Provide report mechanism to let users report infringed problems. Log usage and events in the database so that administrators can trace back such violation.

5. Validation and Verification Process for Analysis Modeling

Having created analysis models of the system, we validated and verified the consistency between each of them to minimize possible conflicts that might occur in the next design phase. Aspects of validation and verification that we focus are listed below:

5.1. Validation and Verification of the same type of Analysis Models

5.1.1. Functional Models

5.1.1.1. Every functional requirement must be addressed as use cases in the use case diagrams.

5.1.1.2. Every use case that is shown in detail by an activity diagram or a use case description must be in the use case diagram.

5.1.1.3. Every actor that exists in use case diagram, activity diagram, and use case description must be consistent.

5.1.2. Structural Models

5.1.2.1. Classes and methods in the class diagram must satisfy all requirements.

5.1.2.2. Attributes and methods in CRC cards must exist in the class diagram.

5.1.2.3. Every collaborator in CRC cards must appear as an association between classes in the class diagram.

5.1.2.4. The responsibility of a class in CRC cards must correlate to the requirements and methods in the class diagram.

5.1.3. Behavioral Models

5.1.3.1. Sequence diagram, communication diagram, behavioral state machine diagram, and CRUDE matrix must be consistent.

5.1.3.2. Messages and manipulation of between any two objects must correlate to their relationship in the CRUDE matrix.

5.2. Validation and Verification between different types of Analysis Models

5.2.1. Functional and Structural Models

5.2.1.1. Classes together with their methods and attributes must be able to satisfy every use case in the use case diagram with the specified flow defined in activity diagram and use case description.

5.2.1.2. Object nodes in an activity diagram must correlate with the methods of the participating classes in that use case.

5.2.2. Structural and Behavioral Models

5.2.2.1. All classes and objects appearing in behavioral models must exist in the class diagram.

5.2.2.2. Operations that a class can perform must correlate to the interaction between classes described in sequence diagrams, communication diagrams, and CRUDE matrix.

5.2.3. Behavioral and Functional Models

5.2.3.1. The sequence of actions shown in sequence diagrams must be the same as those in activity diagrams and use case descriptions.

5.2.3.2. Actors in use case diagram must be presented in the CRUDE matrix.

5.2.3.3. Message passing between classes described in behavioral models must correlate to the flow in use case descriptions and activity diagrams.

6. System Architecture

6.1. Physical Architecture

The system physical will be implemented using 3-Tier architecture, which includes clients, web-servers, and databases. The client's side software is responsible for the application logic and application workflow for the client. For the second tier, web servers are responsible for performing data access and data processing before the data is sent to the client and database. This tier is also known as back-end services for web services. The last tier is the database tier. In this tier, the database partition and replication is set up so that the system can serve thousands of people at the same time. The overview of the system physical architecture is shown in Figure 6-1 below.

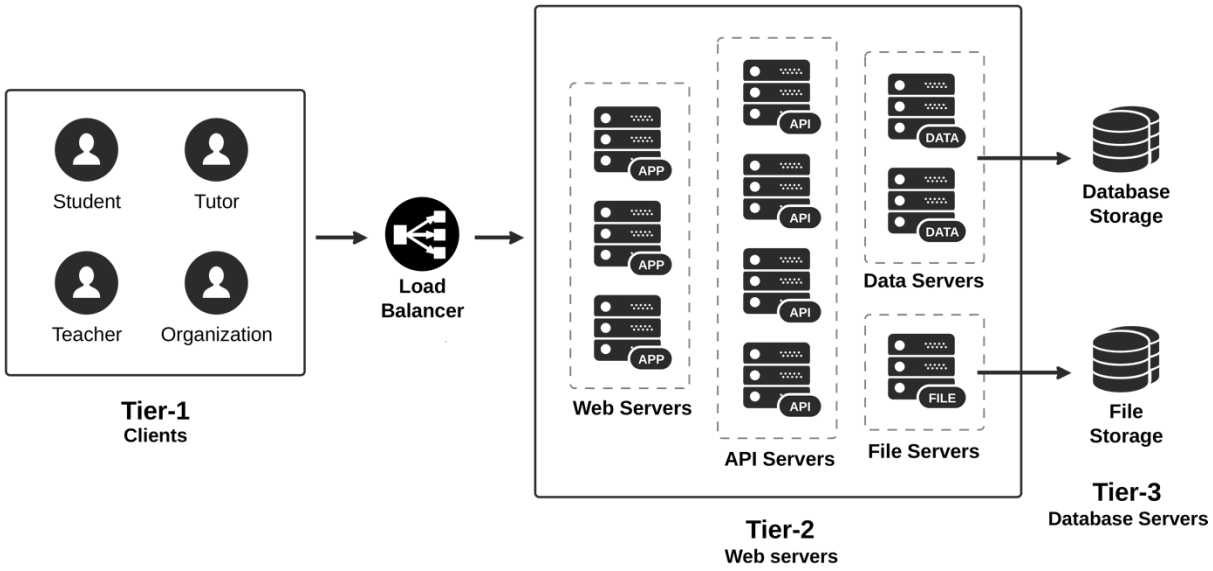

Figure 6-1: Overview of the system's physical architecture

6.2. Network Model

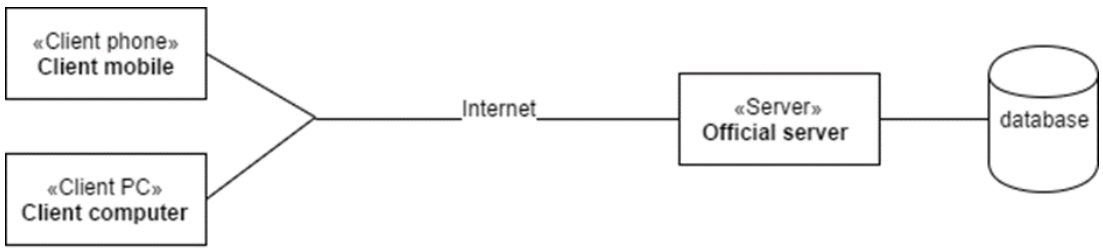

Figure 6-2: Deployment diagram

7. Hardware and Software Specification

Table 7-1: Hardware and software specification

Specification	Standard Client	Web/API Server	File Servers	Database Server
Operating System	Windows, OSX, Ubuntu, Android, iOS	Ubuntu 16.04.2 x64	Ubuntu 16.04.2 x64	Ubuntu 16.04.2 x64
Special Software	Chrome, Safari, Firefox, Edge, Internet Explorer 10, Opera	Nginx, Docker, Docker Compose, Docker Swarm, Node.js	Nginx, Node.js	Docker, Docker Compose, Docker Swarm, Redis, MongoDB
Hardware	Intel Core i3 4 GB / 2 CPUs for 10 GB free disk drive (Desktop) 1 GB / 2 CPUs 1 GB free storage (Mobile)	2 GB / 2 CPUs 40 GB SSD disk	2 GB / 2 CPUs 100 GB SSD disk	2 GB / 2 CPUs 40 GB SSD disk
Network	1–10 Mbps	300 Mbps	300 Mbps	300 Mbps