A Descriptive Design for a Smart Kitchen Management Application (SKM)

Pita Jarupunphol*, Wipawan Buathong†, Tanagrit Chansaeng‡, Nasith Laosen §
Department of Information Technology, Phuket Rajabhat University
Phuket, Thailand
e-mail: ∗p.jarupunphol@pkru.ac.th, †w.buathong@pkru.ac.th, ‡tanagrit.c@pkru.ac.th, §n.laosen@pkru.ac.th

***Abstract*—This research aimed to design a mobile application for smart kitchen management (SKM) for reducing the number of purchased kitchen items/ingredients derived from ignorant and unorganised behaviour. The UML (Unified Modeling Language) and Z schemata were used in a design process to represent discourses associated with the system. The logical arguments defined in the SKM state schema and operation schema enforce essential rules regarding how the SKM should be designed. The SKM was also experimented with 180 participants, who were required to answer the questionnaire comprising question items adapted from USE Questionnaire for measuring four usability dimensions of the SKM (i.e., usefulness, ease of use, ease of learning, and satisfaction). The experimental results showed that the application could function properly and had potential to address the number of unnecessary purchases of kitchen items/ingredients and also prioritised the importance of such items/ingredients. The participants were satisfied with the application and perceived that the application was easy to use and useful for kitchen management allowing them to prepare a list of 'to be purchased' items/ingredients more efficiently.**

Keywords-application; smartphone; smart city; smart kitchen; Z-notation

I. INTRODUCTION

Phuket and Chiang Mai are driven by the Ministry of Information and Communication Technology (ICT) as Thailand's two leading smart cities [1]. According to IEEE [2], a smart city brings together technology, government and society to enable six key characteristics, including smart economy, smart mobility, smart environment, smart people, smart living, and smart governance. Smart living, in particular, is one of the strategic concept of the smart city development with multiple ICT and Internet of Things (IoT) solutions to provide a better quality of health, well-being and happiness, and automation. In addition, QR Code (Quick Response Code) is a matrix barcode, which was initially designed in Japan but nowadays has been widely supported by almost all of the smartphones. The QR code has been widely applied in preference to a barcode due to its faster readability, greater storage, and greater distance [3]. Users with a smartphone equipped with a QR code reader can scan the QR code to display text, open a web page, or retrieve geographical information via GPS.

This article proposes a smartphone application based on the use of QR Code to manage raw materials in a household kitchen namely smart kitchen management (SKM) in which users can add, delete, and modify the number of products used in the kitchen. The SKM allows users to select the storage location, including refrigerator or kitchen. While people usually do not monitor or plan what should be purchased, the application also generates a report representing the items/ingredients about to run out by using the red colour indicator. The SKM will facilitate users in managing the kitchen such as add, delete, check the list of kitchen appliances. The SKM also provides basic, but essential information for the user, e.g., the priority in the purchase of kitchen appliances, reminders, and preliminary cost estimations. The execution of SKM will experiment in the cloud for cost-saving.

II. LITERATURE REVIEWS

According to Business Innovation Observatory conducted by European Union [4, p.3], "Smart Living is a trend encompassing advancements that give people the opportunity to benefit from new ways of living. It involves original and innovative solutions aimed at making life more efficient, more controllable, economical, productive, integrated and sustain- able". In this section, smart living inventions related to the notion of SKM are discussed below.

A. Previous Works

A smart refrigerator is one of smart living technologies that allows users to order food or ingredients from the store via a touch screen interface. For example, Samsung smart refrigera- tor integrates IoT and an online payment option developed by MasterCard in which the user can make a payment via credit card [5]. Besides, the smart refrigerator has an internal camera inside the fridge making it possible to check the quantity of food or ingredients in the fridge while the user is out. This innovation is driven by Tizen operating system, which is an open operating system built by a community of developers to address different industry requirements [6]. However, the user is required to purchase a new refrigerator if he/she wants to experience this smart living technology in daily basis. In addition, the current price of this innovation is around $6,000, which is still not affordable by many home users [7].

978-1-5386-5383-8/18/$31.00 ©2018 IEEE

'Tooyen' is an application for a refrigerator expired food alert in which the user can also assess raw materials in the fridge about how long they can be kept [8]. It is an application designed for address issues associated with the loss of forgetting items in the refrigerator when they are still in good conditions. Tooyen will help the user determine the expiration date of a number of items stored in the refrigerator and ranked for the user.

'Best Before' is a food expiry date notification application developed by a student, who intends to save his/her money and the environment from unnecessarily waste food [9]. The application was designed for both iOS and Android operating systems in which basic information, including expiry date of the food must be stored in the application. The product barcode can be scanned using a smartphone's camera and added to a list of food. Best Before will alert the user when the stored food in the fridge are close to expiration. The alerts can also notify the expiration of other goods in addition to food.

SKM is different from applications reviewed above, since it was designed to facilitate the users in managing the kitchen materials or items at a low cost. The user can check, add, delete, and prioritize the list of kitchen ingredients. The user can also set a reorder point for the ingredients commonly used in the kitchen.

III. METHODOLOGY

The methodology includes different procedures commencing from SKM analysis and design to SKM testing.

A. SKM analysis and Design

The SKM was analysed and designed using the UML (Unified Modeling Language) use case diagram [10]. Figure 1 represents SKM use case diagram containing two major actors: 1) administrator (the person responsible for generating QR code, adding kitchen item details to the database name, including item name, item image, product unit, and the gen- erated QR code); and 2) user (the person who can retrieve the product details from the system and keep them on his/her mobile phone. The product details will be stored in the form of a text file. When the user scans the QR Code, SKM will retrieve the item information and allow the user to make the decision whether to keep the item. The user can check the value of the item for his/her decision-making process. In addition, schemata of the Z notation [11], [12], a formal specification language named after Zermelo–Fraenkel set theory, was also used to formalize core elements involved in the SKM. In this case, two major state schemata and operation schemata are classified into administrator and user.

1) Administrator: Administrator is the person who plays a crucial role in adding, deleting, and editing details of ingredients in SKM. As such, there are some functionalities of the SKM limited to the administrator. The *SKMadmin* state schema is used to declare a static view of the SKM related to the admin. The upper part of the schema consists of variable declarations of the *SKMadmin*. *SYSTEM*, and *INGREDIENT* are two basic type definitions where 1) system is a variable of $\mathbb{P}$ *SYSTEM*; and 2) *ingredient* is a variable of $\mathbb{P}$ *INGREDIENT*. The storage partial function from SYSTEM to *INGREDIENT* is used to represent a relationship that maps each element of SYSTEM to one element of *INGREDIENT* at most.

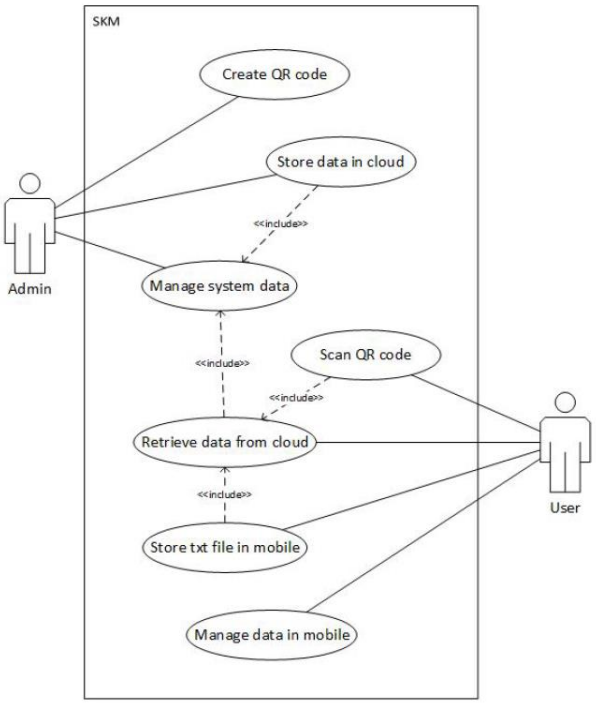

Figure 1. The SKM use.

The lower part of the schema consists of logical statements which define the operation. The dom function enforces that the domain of relation R is the set of elements in X related to some elements in Y ($\mathrm{dom}\ R = \{x : X; y : Y \mid x \mapsto y \in R \bullet x\}$). In the meantime, the ran function also posits that the range of R is the set of elements of Y to which some elements of X are related ($\mathrm{dom}\ R = \{x : X; y : Y \mid x \mapsto y \in R \bullet y\}$). In this case, the set system is to be a subset equal to the domain of the storage function while the set *ingredient* is to be a subset equal to the range of the storage function.

SKMadmin

$$system : \mathbb{P}\ SYSTEM$$
$$ingredient : \mathbb{P}\ INGREDIENT$$
$$storage : SYSTEM \nrightarrow INGREDIENT$$

$$system \subseteq \mathrm{dom}\ storage$$
$$ingredient \subseteq \mathrm{ran}\ storage$$

Since the administrator can add, delete, and edit kitchen ingredients in the application, the initialization schema $ADMIN_{INIT}$ is created to set an initial stage before an admin starts adding an ingredient to a system. Two variables,

including *system* and *ingredient* are set at ∅ in the first stage when the ingredient has not yet been added to the system. The *SKMadmin* in the upper part of the schema implies that all variable declarations and logical statements defining the operation in the *SKMadmin* state scheme will be included in the initialization schema $ADMIN_{INIT}$

$$
\begin{array}{|l}
\hline
ADMIN_{INIT} \\
SKMadmin \\
\hline
system = \emptyset \\
ingredient = \emptyset \\
\hline
\end{array}
$$

The *AdminAdd* operation schema includes the state schema Δ*SKMadmin*. The schema *SKMadmin* in this sense will be used with both its declarations and predicates. The operation schema *AdminAdd* updates the component storage in the schema *SKMadmin*. The upper part of the schema consists of variable declarations in which *s?* (defined as SYSTEM type) and *i?* (defined as *INGREDIENT* type) are two emerging elements (or input arguments) in *system* and *ingredient* respectively. The lower part of the schema consists of logical statements which must hold when the operation starts. The add ⊆ storage defines that add activity is always covered by storage operation. In other words, the storage function will be updated when the ingredient is added to the system (∀ *s?* ↦ *i?* ∈ add • *s?* ↦ *i?* ∈ storage).

The predicate system′ = system ∪ *s?* ↦ *i?* is post-condition of add, used to represent an element *s?* has been updated in system (*system′ = system ∪ a?*) and an element *i?* has been updated in *ingredient* (*ingredient′ = ingredient ∪ i?*) by add. In other words, the set of ingredients will be augmented with the new ingredient to the system.

$$
\begin{array}{|l}
\hline
AdminAdd \\
\Delta SKMadmin \\
s? : SYSTEM \\
i? : INGREDIENT \\
add : SYSTEM \pfun INGREDIENT \\
\hline
add \subseteq storage \\
system' = system \cup \{s? \mapsto i?\} \\
\hline
\end{array}
$$

The *AdminDelete* operation schema also includes the state schema Δ*SKMadmin*. The delete ⊆ storage in the lower part defines that delete activity is a subset of storage. The predicate *system′ = system \ s? ↦ i?* is post-condition of delete, used to USER represent elements *s?* and *i?* that have been updated in *system* and *ingredient* respectively (system′ = system \ *i?*) by delete.

$$
\begin{array}{|l}
\hline
AdminDelete \\
\Delta SKMadmin \\
s? : SYSTEM \\
i? : INGREDIENT \\
delete : SYSTEM \pfun INGREDIENT \\
\hline
delete \subseteq storage \\
system' = system \setminus \{s? \mapsto i?\} \\
\hline
\end{array}
$$

The *AdminUpdate* operation schema is used to update the ingredient. The state schema Δ*SKMadmin* is included in the schema because declarations and predicates of the *SKMadmin* will be used. The predicate *system′ = system* ⊕ *s? ↦ i?* is post-condition of update, used to represent an element *i?* that has been updated in ingredient (*system′ = system* ⊕ *i?*) by *s?* of the update function.

$$
\begin{array}{|l}
\hline
AdminUpdate \\
\Delta SKMadmin \\
s? : SYSTEM \\
i? : INGREDIENT \\
update : SYSTEM \pfun INGREDIENT \\
\hline
update \subseteq storage \\
system' = system \oplus \{(s?, i?)\} \\
\hline
\end{array}
$$

2) User: User is the common user who is capable of using the SKM application. The SKMuser state schema declares a static view of the SKM related to the user. *INGREDIENT* and ℕ are two basic type definitions where 1) *ingredient* is a variable of ℙ *INGREDIENT* ; and 2) amount is a variable of ℕ. The statusamount partial function from *INGREDIENT* to ℕ is used to represent a relationship that maps each element of INGREDIENT to one element of *AMOUNT* at most. The *dom* function in the lower part enforces that the set *ingredient* is to be a subset equal to the domain of the *statusamount* function while the set amount is to be a subset equal to the range of the *statusamount* function.

$$
\begin{array}{|l}
\hline
SKMuser \\
ingredient : \mathbb{P}\, INGREDIENT \\
amount : \mathbb{N} \\
status : INGREDIENT \pfun AMOUNT \\
\hline
ingredient \subseteq \mathrm{dom}\, status \\
amount \subseteq \mathrm{ran}\, status \\
\hline
\end{array}
$$

The initialization schema *USERINIT* is an initial stage before a user starts adding an ingredient to a system. A variables status is set at ∅ in the first stage when the ingredient amount has not yet been connected. All variable declarations and logical statements in the *SKMuser* state scheme will be used in the initialization schema $USER_{INIT}$. To

ensure that the user actions are different from the actions reserved for the administrator, *UserAdd*, *UserRemove* and *UserSearch* are three operational schemata reserved for the user.

$$USER_{INIT}$$
$$SKMuser$$
$$status = \emptyset$$

The operation schema *UserAdd* includes the state schema $\Delta SKMuser$ and relevant declarations and predicates. The operation schema *UserAdd* will update the component status in the schema *SKMuser*. The variable declarations a? (defined as *AMOUNT* type) and *i?* (defined as *INGREDIENT* type) are two input arguments in amount and ingredient. The add $\subseteq$ status defines that the status in the state schema $\Delta SKMuser$ will be updated when the amount is added to the ingredient. The predicate $ingredient' = ingredient \cup \{i? \mapsto a?\}$ is post- condition of add, used to represent an element a? has been updated in amount ($amount' = amount \cup a?$) and an element *i?* has been updated in ingredient ($ingredient' = ingredient \cup i?$) by add.

$$UserAdd$$
$$\Delta SKMuser$$
$$i? : INGREDIENT$$
$$a? : AMOUNT$$
$$add : INGREDIENT \nrightarrow AMOUNT$$
$$add \subseteq statusamount$$
$$i? \notin \mathrm{dom}\, add$$
$$a? \notin \mathrm{ran}\, addingredient' = ingredient \cup \{i? \mapsto a?\}$$

The operation schema *UserRemove* includes the state schema ΔSKM and relevant declarations and predicates. The remove $\subseteq$ status defines that remove activity is a subset of status. The predicate $ingredient' = ingredient \{i? \mapsto a?\}$ is post-condition of remove. This implies that elements *i?* and a? have been updated in ingredient and amount.

$$UserRemove$$
$$\Delta SKMuser$$
$$i? : INGREDIENT$$
$$a? : AMOUNT$$
$$remove : INGREDIENT \nrightarrow AMOUNT$$
$$remove \subseteq status$$
$$i? \in \mathrm{dom}\, status \land a? \in \mathrm{ran}\, status$$
$$ingredient' = ingredient \{i? \mapsto a?\}$$

The *UserSearch* is an observation schema that includes the state schema $\Xi SKMuser$ to represent the user's searching to- ward the SKM with the postcondition remain unchanged. The operation schema *UserSearch* will not update the component *ingredient* and *amount* in the schema *SKMuser*. The predicate $ingredient' = ingredient \cup \{i? \mapsto a!\}$ represents a condition when the ingredient is searched by the user, but no operations will update the ingredient at this stage.

$$UserSearch$$
$$\Xi SKMuser$$
$$i? : INGREDIENT$$
$$a! : AMOUNT$$
$$search : INGREDIENT \nrightarrow AMOUNT$$
$$i? \in ingredient$$
$$a! \in amount$$
$$ingredient' = ingredient \cup \{i? \mapsto a!\}$$

B. SKM Development

SKM was designed for being compatible with Android 4.4 or higher with the Internet connection to scan QR code. Since SKM was built for the Android operating system, there are two primary tools used to develop SKM: 1) Android Studio (the official integrated development environment (IDE) designed for Android development) and 2) Firebase Storage (a realtime database with an API allowing developers to store and sync images, audio, video, or any generated content across multiple clients).

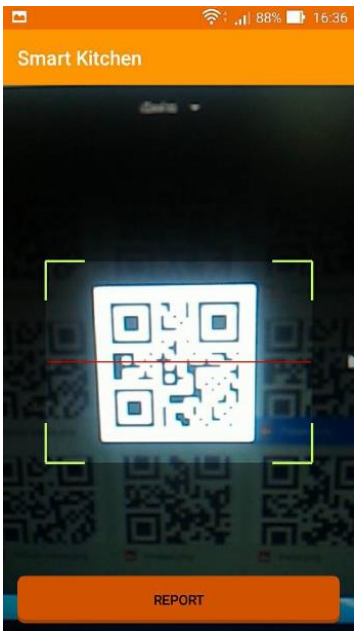

Figure 2. The SKM QR code scanning screen.

Figure 2 represents a QR code scanning screen where a user can scan QR code or select report option to check the status of ingredients. Figure 3 illustrates that the user can turn the 'add' mode on or 'use' if he/she chooses to scan QR code.

Figure 3. The SKM add or use screen.

In Figure 4, SKM will display QR codes of ingredients and a report of the total number of available ingredients in sequence. Please note that SKM allows the user to customize a reorder point and the ingredient amount will be displayed in red if the amount has reached the reorder point. The ingredients can also be displayed based on storage locations.

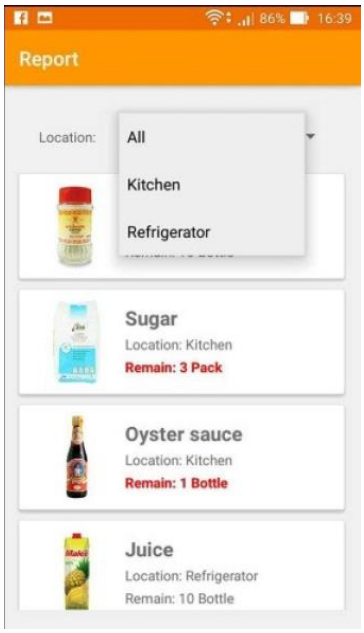

Figure 4. The SKM report screen.

C. SKM Testing

The questionnaire comprises question items adapted from USE Questionnaire developed by Lund [13], who adjusted the questionnaire for measuring usability from time to time. Initially, ease of use, usefulness, and satisfaction were proposed by the author as three main dimensions discriminating between interfaces with partial correlations among them. Improving ratings of one dimension can increase those of the others. The questionnaire is based on seven-point Likert scales ranging from strongly disagree to strongly agree. According to the USE current version, there are thirty question items used to evaluate user attitudes towards four dimensions of usability, i.e., ease of use, ease of learning, usefulness, and satisfaction [14]. In addition to the question items, a comment box is also provided at the end of the questionnaire for the participants, who may have suggestions to improve SKM. Some examples of adapted question items for this research are below

IV. Results and Discussion

After the SKM was experimented with all of the 180 participants, there were no errors with the SKM designed functionalities. The participants were asked to test with four items located in kitchen area and refrigerator. The ingredients could be placed or removed after they had been scanned by the participants. In terms of the results derived from the questionnaire, different question items were classified into their appropriate types for attitude measurement, including usefulness, ease of use, ease of learning, and satisfaction. In detail, ease of learning is the highest rank of user attitudes due to the highest mean ($\mu = 5.89$). In addition, the lowest standard deviation implies that most of the numbers are very close to the average ($\sigma = 0.74$). The users are also very satisfied with the SKM ease of use ($\mu = 5.70, \sigma = 0.88$) and usefulness ($\mu = 5.44, \sigma = 0.99$). Satisfaction is the lowest rank because of the lowest mean and the highest standard deviation in comparison with others ($\mu = 4.87, \sigma = 1.11$). This implies that the numbers in this dimension are spread out. Table I shows the experimental results regarding SKM usability in different aspects.

TABLE I. The SKM Usability Experimental Results

Descriptive Statistics

	Ease_of_Use	Ease_of_Learning	Usefulness	Satisfaction
Valid	180	180	180	180
Missing	0	0	0	0
Mean	5.703	5.894	5.440	4.871
Std. Deviation	0.8847	0.7380	0.9944	1.106
Minimum	2.200	3.670	2.170	1.330
Maximum	7.000	7.000	7.000	7.000

The results in Table II show that there is a significant positive relationship between ease of use and usefulness ($r = .675$) as well as ease of use and satisfaction ($r = .631$). In particular, the highest positive relationship exists between usefulness and satisfaction ($r = .728$). These correlations are consistent with the USE questionnaire developer [13], who stated that all these three dimensions are positively correlated with each other. When ease of use, usefulness, and satisfaction are measured with ease of learning, all of them have minimal negative relationships with ease of learning at r =

-.112, -.140, -.138 respectively. This also implies that these three dimensions are still in the same direction when tested by ease of learning.

TABLE II. CORRELATIONS OF FOUR DIMENSIONS ADAPTED FROM THE USE QUESTIONNAIRE

Pearson Correlations

		Ease_of_Use	Ease_of_Learning	Usefulness	Satisfaction
Ease_of_Use	Pearson's r	—	-0.112	0.675***	0.631***
	p-value	—	0.133	< .001	< .001
Ease_of_Learning	Pearson's r		—	-0.140	-0.138
	p-value		—	0.061	0.066
Usefulness	Pearson's r			—	0.728***
	p-value			—	< .001
Satisfaction	Pearson's r				—
	p-value				—

* p < .05, ** p < .01, *** p < .001

V. CONCLUSIONS AND FUTURE WORK

This article introduces SKM, which is a smartphone application developed for managing the household kitchen. The application could potentially address several problems in the purchase of kitchen items/ingredients long left in the fridge or kitchen through a smartphone application. SKM is capable of facilitating the priorities for the purchase of kitchen items/ingredients. This application can be an alternative aspect of kitchen management in which people could monitor how many items/ingredients left in the refrigerator or kitchen.

Given that automation is an essential factor driving smart technologies for many aspects of smart city, the next version of SKM should be equipped with more automatic features. The participants suggested that the application should provide more features fulfilling the notion of smart kitchen management despite the questionnaire positive results. The suggestions were related to potential functional and non-functional requirements of SKM. For example, the application should also be designed for iOS in addition to Android. The application should also calculate and indicate the reorder point when the items/ingredients were about to expire besides to be purchased. The application should provide automatic calculation of the average number of days before the item will expire. Since the QR code experimented in this research was purposely generated for the experimentation only. the application should be compatible with those QR codes standardized and generated by retailers. The application should provide the user with turn on and off push notifications for a single or all ingredients. Primarily, the application should allow the user to order ingredients from major merchandise retailers providing online shopping and delivery services in Thailand.

REFERENCES

[1] W. Buathong, P. Jarupunphol, and T. Chansaeng, "A healthy food rec- ommendation application (hfra) for smart eating," Journal of Industrial Technology Ubon Ratchathani Rajabhat University, vol. 7, no. 1, pp.1–11, January - June 2017.

[2] "About IEEE Smart Cities," IEEE Smart Cities, 2016. [Online]. Available: http://smartcities.ieee.org/about

[3] P. Jarupunphol, W. Buathong, and T. Chansaeng, "The MTAS for addressing motorcycle theft," in The 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON 2016), Chiang Mai, Thailand, June. 2016, pp. 1–6.

[4] L. Probst, E. Monfardini, L. Frideres, D. Cedola, and P. Luxembourg, "Smart living," European Union, Business Innovation Observatory, February 2014. [Online]. Available: https://ec.europa.eu/docsroom/documents/13407/attachments/2/translations/en/renditions/native

[5] "MasterCard, Samsung Make Everyday Shopping Easier in Tomorrows Smart Home with Launch of Groceries by MasterCard App," MasterCard Press Releases, January 2016. [Online]. Available: https://newsroom.mastercard.com/press-releases/mastercard-samsung-make-everyday-shopping-easier-in-tomorrows/smart-home-with-launch-of-groceries-by-mastercard-app/

[6] R. Messier, "Chapter 12 - newer technologies," in Operating System Forensics. Syngress, 2016, pp. 331 – 349.

[7] "MasterCard Releases Its IoT App for Samsung Refrigerator," BankInnovation, May 2016. [Online]. Available: http://bankinnovation. net/2016/05/mastercard-releases- its-iot- app-for- samsung-refrigerator/

[8] "Tooyen – an application for a refrigerator expired food alert (in Thai)," thaiware.com, 2015. [Online]. Available: http://software.thaiware.com/7186- Tooyen-App.html

[9] "Best Before," Google Play, May 2016. [Online]. Available: https://play.google.com/store/apps/details?id=com.nicedistractions.bestbefore&hl=en

[10] D. Kulak and E. Guiney, Use Cases: Requirements in Context, 2nd ed. Boston: Addison Wesley, 2003.

[11] J. Jacky, The Way of Z: Practical Programming with Formal Methods. Cambridge: Cambridge University Press, 1997.

[12] J. M. Spivey, The Z Notation: A Reference Manual. University of Oxford, 1998.

[13] A. M. Lund, "Measuring usability with the use questionnaire," Usability Interface, vol. 8, pp. 3–6, 2001. [Online]. Available: http://www.stcsig.org/usability/newsletter/index.html

[14] "USE questionnaire: Usefulness, satisfaction, and ease of use," garyperlman.com. [Online]. Available: http://garyperlman.com/quest/quest.cgi?form=USE