Scientific Journal of Impact Factor (SJIF): 5.71

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 5, Issue 04, April -2018

CLOUD-BASED ARCHITECTURE OF A SMART EXPIRY SYSTEM WITH IOT DEVICE

Tareq Khan

School of Engineering Technology, Eastern Michigan University, Michigan, Ypsilanti, United States

Abstract — *Wasting of food is a major problem worldwide. One of the reasons for this food waste is that the customer forgets the expiry date of the food after purchase and do not consume the food before it expires. When the customer finds out that the food is expired, it is tossed out. Moreover, consuming expired foods may cause serious health issues. In this project, a novel cloud-based smart expiry system is developed which sends an automatic notification to customer's smartphone and on an Internet of Things (IoT) display device (that can be attached to a refrigerator) several days before the food expires. In the proposed method, the customer gives a customized smart-expiry magnetic card to the checkout operator for swiping. The checkout operator app generates a table containing product names with expiry dates and it is uploaded to the cloud. The customer's smartphone app then automatically downloads the table from the cloud and the customer is all set to receive the expiry date notifications. Alternatively, in case the customer forgets or does not have the smart-expiry card with him/her in the store, a Quick Response (QR) code is printed on the receipt. The customer just scans the single QR code with his/her smartphone and the table is downloaded from the cloud to the phone. Compared with other works, the proposed method does not need tedious manual entry of each product name and expiration date into the smartphone app, rather providing a smart-expiry card to the checkout operator or scanning a single QR code is sufficient. A prototype of the proposed system has been developed and tested.*

Keywords- *Internet of Things, Cloud, QR code, Android Things, Raspberry Pi, Expiry date, Notification*.

I. INTRODUCTION

In the United States, 40% of the food supply is wasted in each year. It costs around $218 billion per year for producing, transporting, processing, and disposing of this unused food, and two-thirds of this lost economic value are due to household food waste [1][4]. One of the reasons of this food waste is that the consumer forgets the expiry date of the food after purchase and do not consume the food before it expires, and when he/she finds out that the food is expired, it is tossed out. It is also difficult to manually check the expiry dates of all the items stored in a refrigerator full of foods and keep in mind all the expiry dates. Also, consuming foods that have expired may cause serious health problems [5][6]. In this project, a novel cloud-based smart expiry system is developed which sends an automatic notification to customer's smartphone and on an Internet of Things (IoT) display device (that can be attached to a refrigerator) several days before the food expires. These notifications will aware the customers of the recently expiring foods and will give peace of mind. In the proposed method, the customer gives a customized smart-expiry card to the checkout operator for swiping. The checkout operator app generates a table containing product names with expiry dates and it is uploaded to the cloud. The customer's smartphone app then automatically downloads the table from the cloud and the customer is all set to receive the expiry date notifications. Alternatively, in case the customer forgets or does not have the smart-expiry card with him/her in the store, a Quick Response (QR) code is printed on the receipt. The customer just scans the single QR code with his/her smartphone and the table is downloaded from the cloud to the smartphone. No manual entry of each of the product name or expiration date by the customer into the smartphone is required.

The significance and advantages of the smart expiry system are mentioned below.

- American households waste 14% of their food purchases on an average and 15% of that includes products that never opened. According to [2][3], an average family of 4 tosses out $590 per year, making it a serious economic problem. One way to reduce this waste is to consume the food before it's expiry date. The proposed smart expiry system will send automatic reminders several days before the food is going to expire and thus ensure that item is consumed or donated, and not wasted.
- Consuming expired food may cause minor to serious health problems [5][6]. Eating expired food may trigger food poisoning causing symptoms such as vomiting, fever, cramping in the stomach area, dizziness, diarrhea, and dehydration. Some food stored past its use-by date in poor conditions can even become contaminated with the serious bacterial infections, molds, and mycotoxins. Mycotoxins can cause itchiness, nausea, dizziness and sometimes headaches. The proposed smart expiry system will inform when the food is about to expire and when it is unhealthy to eat, ensuring better health for the family.
- Taste of some foods may degrade when they reach near the expiry date [5]. The automatic notification will alert the consumer several days before the expiry date so the food is consumed when it has the best taste.

- The proposed could-based smart expiry system can also be used in items other than food such as medicine, toiletry, and cosmetics.
- The expiry date labels in products contain prefixes such as “expiry,” “sell by,” “best before,” “use by” etc. Customers sometimes misinterpret these labels and throw away food once the date is passed because they mistakenly think the date is an indicator of safety, however, for most foods the date is the manufacturer’s best guess of how long the product will be at its best quality. As reported in [1], standardizing date labeling is the most effective solution for decreasing food waste, and could save tons of food that is wasted each year. In this proposed smart expiry system, the expiry date will be written using barcode along with human readable English. The barcode can contain several dates with exact prefix information coded in numbers indicating “best before”, “unsafe” etc. This research will reinforce the standardization of the expiry date as proposed by [1], which can help to reduce the significant amount of food that consumers unnecessarily discard.
- By using the proposed smart-expiry system, the consumer can better manage what items to cook or consume, and also when the item needs to be purchased again.
- The system will also cause more sells for the grocery stores, as the customers will consume the item quicker and will come to the store for next purchase more frequently.

II. RELATED WORK

Several related works are found in the literature about the expiry date. The work in [13] proposes a solution that helps the visually impaired to identify a product and subsequently recognize the expiration date on a product using a handheld smartphone. However, in this method, each product needs to be scanned one by one, and it does not provide automatic reminders. The expiry date is recognized by Optical Character Recognition (OCR) which has a high probability of error due to non-standard format, font, and image quality. Moreover, the app needs to contain a table that maps barcode to product name for all available products, which is difficult because there are millions of products and will take large storage space. Samsung recently introduced Family Hub refrigerator [16], which has 3 cameras to take images of items inside the refrigerator and a large display unit connected to the refrigerator. The user can manually set expiry dates of the items that are visible by the cameras using drag-drop reminder notes. This method involves manual entry and cannot be used in a crowded fridge where many items are not visible by the front cameras. A concept of QR fridge magnets to keeps track of food’s expiration dates is reported in [10]. Each QR fridge magnet device is used to track one product’s expiry date only and the device has a processor, display, and QR code scanner hardware. This idea involves purchasing of many QR fridge magnet devices, which can be expensive for the customer.

Several apps have been recently developed to generate reminders for expiry date. In the app in [11], the user needs to manually enter the product name, purchase date, expiration date, and it generates reminders several days before the expiration date. However, this manual entry of each purchased product’s name and expiration date are tiresome, time-consuming, and not efficient. The app in [12] attempts to develop an easier way for entering the product information by only scanning the barcode of the product using the smartphone’s camera. However, the table for converting barcode to product name is not available in the app, rather the table needs to be manually created by the user for all the products he/she uses, which is an overhead.

Compared with these related works, the proposed approach seeks to remedy the tiresome manual entry of each product information. The user just provides a smart-expiry card to the checkout operator or scans a single QR code printed on the receipt using his/her smartphone. Also, unlike the works in [12][13], the proposed approach does not need to store the barcode to product name conversion table of all available products in customer’s phone, rather the barcode to product name conversion is done with the help of the store’s product table. Thus small storage space is required for the smartphone app.

III. SYSTEM ARCHITECTURE

The proposed smart expiry system is shown in Fig. 1. The expiry date is generally written using the English language on a product. In the proposed method, the expiry date is written using bar code as shown in Fig. 1(a) along with human readable English. After the customer chooses the product and goes for check-out, the bar code for the product and also the barcode for the expiry date is scanned by the checkout operator, as shown in Fig. 1(b), using a barcode reader as shown in Fig. 1(c). This step is repeated for all the purchased products which have expiry dates. The app used by the checkout operator’s desktop computer, as shown in Fig. 1(d), decodes the barcodes, gets the product names from the store’s barcode to product name conversion database, and creates a new table which contains the purchased product names and expiry dates. In this proposed method, the customer gives a customized smart-expiry card to the checkout operator for swiping. The software gets the customer’s card ID from the card reader, as shown in Fig. 1 (e) and then uploads the table file (having the same name of the card ID) to a cloud, as shown in Fig. 1(h), using the Internet.

The card ID is configured in the smartphone app of the customer. The customer's smartphone app then automatically downloads the table from the cloud and generates the notifications based on the table contents.

An IoT display device, as shown in Fig. 1(k), also downloads the table from the cloud and displays the list of expiring items. It also shows weather and time gadgets. A motion sensor is also interfaced with the device. When a nearby motion is detected – the device reads out loud the items that are soon expiring - once in a day. The device is designed using Raspberry Pi [17] with Android Things [21] operating system. The device access the Internet using home Wi-Fi.

The main units of the system are briefly described below.

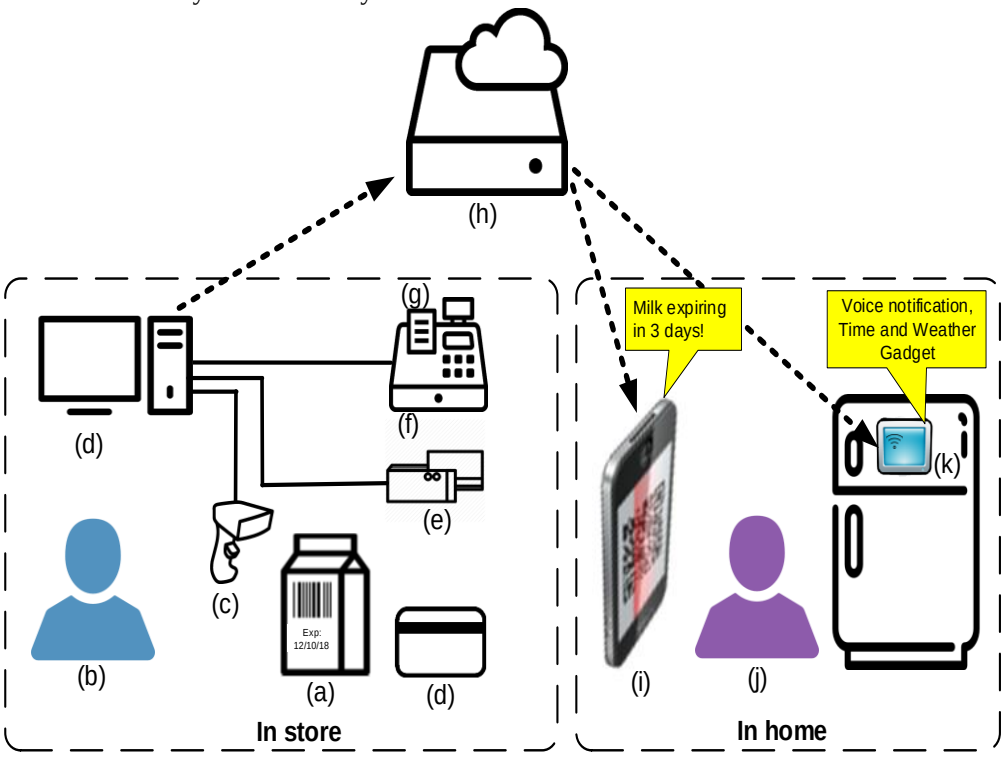


Figure 1. The proposed smart expiry system. In-store (Left), the checkout operator (b) scans product's expiry barcode (a) using a scanner (c); The customer provides a smart-expiry card (d) to the checkout operator for swiping; A table with the product names and expiry dates is created in the computer (d) and the table is uploaded to the cloud (h). In home (Right), the table gets downloaded to the customer's smartphone (i) and to the display IoT device (k) automatically. If no smart-expiry card is swept, the receipt (g), printed by a printer (f), contains a QR code to access the table in the cloud. The customer scans the QR code with the smartphone (i) and the table is downloaded from cloud (h) to smartphone (g) and to the display IoT device (k) automatically.

3.1. Checkout Operator App

When a new customer comes, the checkout operator computer app first generates a unique receipt number, referred as *ReceiptNo*. The format of the receipt number is "*YYYY-ddd-XXXXXXX*" where *YYYY* is the year, *ddd* is the day of the year ranging from 1 to 366, and *XXXXXXX* is a 7 digit running number, ranging from 0,000,001 to 9,999,999. So the system can generate up to 10 million unique receipt numbers daily until year 9999.

The checkout operator scans the barcode of the product and also scans the expiry date of the product if applicable, one by one. A barcode scanner is connected to the checkout operator computer using universal serial bus (USB) port. Keypress events are raised on the computer when the scanner scans barcodes. When a product barcode is scanned, the app gets the product name and its unit price from the store's database and displays them in a list box. When an expiry date barcode is scanned, the preceding scanned product name and the expiry date is appended to a table, *TableExp*. *TableExp* contains the names and expiry dates of all the products that were purchased by the customer. Each record in the table also contains a tick count [24] that works as primary key [25].

In this proposed method, the customer can have a smart-expiry magnetic card. A *CardID* is written on the magnetic card having the format "*SEC-XXXX-XXXX*", where XXXX-XXXX can be an 8 digit unique number. In the case, the customer provides a smart-expiry card to the checkout operator for swapping, the checkout operator app gets the *CardID* from the card reader and downloads the <*CardID*>.*csv* file from the cloud to the local phone. The app then appends the contents of *TableExp* with the <*CardID*>.*csv* file. After the payment is processed, the <*CardID*>.*csv* file is reuploaded to the cloud using the Internet.

In the case, where the customer does not provide a smart-expiry card to the checkout operator, the *TableExp* is saved as *<ReceiptNo>.csv* file. After the payment is processed, the *<ReceiptNo>.csv* file is uploaded to the cloud using the Internet. The checkout operator app generates a QR code image containing the *ReceiptNo*. The error correction capability (ECC) of the QR code is set to level "L" [7]. A purchase receipt is then printed along with the QR code containing the *ReceiptNo.*

3.2. Cloud Storage

The Firebase cloud storage [14] platform is used to store the *<CardID>.csv* and the *<ReceiptNo>.csv* files in the cloud. Firebase storage is a powerful, simple, and an exabyte scale object storage solution. Files can be uploaded, downloaded and deleted using the Firebase software development kit (SDK) functions from the desktop, mobile, and web platforms.
The Firebase SDKs add Google security to file uploads and downloads for Firebase apps. Firebase provides a declarative rules language that allows controlling the read, write, and delete access so that the storage is restricted to only authenticated users. To access data from the Firebase cloud, the smartphone app needs to be registered in the cloud. Thus, it is not possible to make an app by a malicious user and download data from the cloud without having the authentication details.

3.3. Smartphone App

The app is designed for Android platform. The first screen of the app contains a list-view box. Each record in the list-view box shows the name of the item, days remaining to expire or how many days ago it expired, and the actual expiry date. The list-view box reads this information from *expproduct.csv* file that stored on the local phone. It contains the sorted list of items that are going to expire. The different sections of the app are described below.

3.3.1. Configurations. The app contains a settings menu for configuring the smart expiry card ID, synchronization interval with the cloud, and notification. From smart expiry card ID settings, the user can enter the card ID, *CardID*, either by scanning the QR code that is drawn on top of the card or by manually typing the ID in a textbox. As soon as the card is configured, the app uploads *expproduct.csv* file from the local phone to the cloud with the name of *<CardID>.csv* so that the cloud file contains all the expiry date information for that customer that was generated before the customer started using the card. From synchronization interval with cloud settings, the user can specify the interval in minutes, *SyncInterval*, after which data is downloaded from the cloud periodically. From the notification settings, the user sets the days before notification, *DaysBeforeNotification*, and notification hour of the day, *NotificationHour*. The user can also choose notification properties such as sound, vibration, and light.

3.3.2. Downloading data from the cloud.

- **Case 1: Smart-expiry card was used during purchase:** If the user has a smart-expiry card configured in the app, a *service* [15] for downloading the *<CardID>.csv* file after each *SyncInterval* from the cloud to the local phone is automatically started after booting the phone. After the file is downloaded, the contents of the CSV file are read in a *list.* Then the list is sorted based on the number of remaining days to expire from the current date. Bubble sort is used for sorting the records. Then the sorted list is saved in *expproduct.csv* file. From the content of this file, the list-view box on the front screen is updated. Then the downloaded *<CardID>.csv* file is then deleted from the local phone. The flowchart of the download service is shown in Fig. 2(a).

- **Case 2: Smart-expiry card was not used during purchase:** If the customer does not provide the smart-expiry card to the checkout operator for swiping, then the customer gets a receipt where a smart expiry QR code is printed. The smartphone app contains a button on the first screen for scanning the QR code. When the button is pressed, the front camera is turned on for scanning the QR code. A QR code reader library is used in the app. It decodes the code and outputs the message contained in the QR code in a string. The string contains the *ReceiptNo*. Then *<ReceiptNo>.csv* file is downloaded from the Firebase cloud to the local phone.

    - **Case 2.1: Smart-expiry card was not used during purchase and a card is configured in phone:** Now, if any smart-expiry card is configured in the customers smartphone app, and the customer is scanning a QR code that is printed on the receipt, then it means that the customer forgot or did not have the card with him/her during checkout in the store. It may be the case that the customer has just configured a new card information on the phone and now scanning a QR code from an old receipt. In this case, the *<CardID>.csv* file is downloaded from the cloud, the contents of *<ReceiptNo>.csv* file is appended with the *<CardID>.csv* file and then the *<CardID>.csv* file is reuploaded in the cloud. In this way, the system goes to the state as if the card was swept during the purchase. Then the local phone copy of the *<CardID>.csv* and *<ReceiptNo>.csv* files are deleted. The *<ReceiptNo>.csv* file in the cloud is also deleted. To get the immediate effect of the QR code scan, the service for regularly downloading *<CardID>.csv* file is started as shown in Fig 2(a). The flowchart of this case after the QR code is scanned is shown in Fig. 2(b).

- **Case 2.2: Smart-expiry card was not used during purchase and no card is configured in the phone:** Now, if there is no smart-expiry card configured in the customer's smartphone app and the customer is scanning a QR code that is printed on the receipt, then it means that the customer does not have any card. In this case, the content of the *<ReceiptNo>.csv* file is appended to the existing *expproduct.csv* file that is stored on the local phone. Then the *expproduct.csv* contents are sorted based on the number of remaining days to expire from the current date. Then the list-view box on the front screen is updated with the contents of the *expproduct.csv*. The downloaded *<ReceiptNo>.csv* file is deleted from the local phone and also from the cloud. The flowchart of this case after the QR code is scanned is shown in the flowchart in Fig. 2(c).

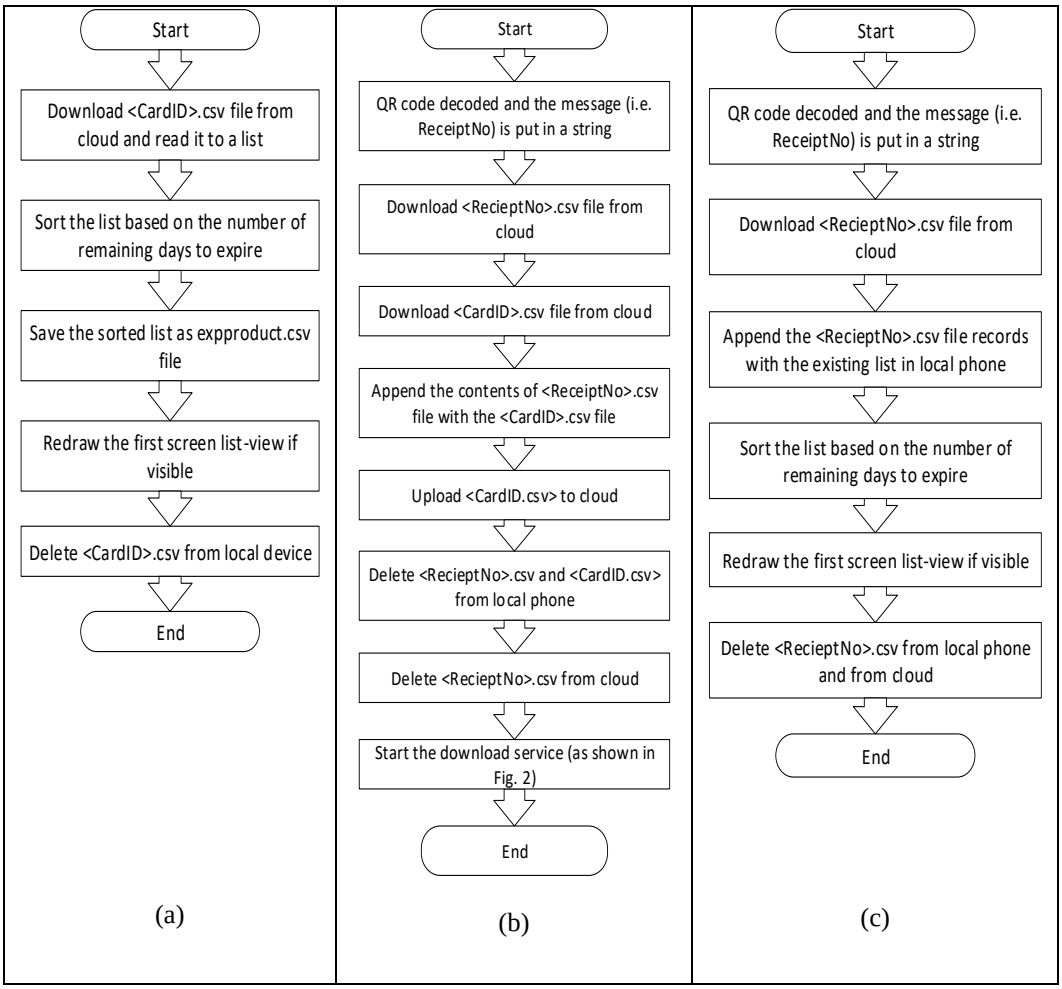


Figure 2. (a) Case 1 - Flowchart of the download service when smart-expiry card was used during purchase; (b) Case 2.1 – Flowchart when the smart-expiry card was not used during purchase and a card is configured in the phone; (c) Case 2.2 - Flowchart when the smart-expiry card was not used during purchase and card is not configured in the phone

3.3.3. Notification service. After booting the phone, a *service* for generating smart expiry notifications is also automatically started. The service checks the expiry dates every day at the *NotificationHour*, that is set by the user in the settings. For each item, the service calculates the days remaining, *DaysRemaining*, by calculating the date difference between the expiry date and the current date. If the *DaysRemaining* is smaller than or equal to *DaysBeforeNotification*, then notification for that item is generated.

3.3.4. Deleting items. After a food is consumed, it needs to be removed from the app so that the app does not generate further notifications. This is done on the first screen by making a long-click on the record of the list-view box. When a long-click is made on a record, the app asks for confirmation to delete. If confirmed by the user, then the selected record is stored in the string *s_del* and the actions shown in the flowchart in Fig. 3 are executed. The record is deleted from the cloud. Thus deleting any item from the smartphone also deletes the item from the IoT device and vice-versa. If the user has no card configured in the smartphone, then the item is deleted from the local phone only.

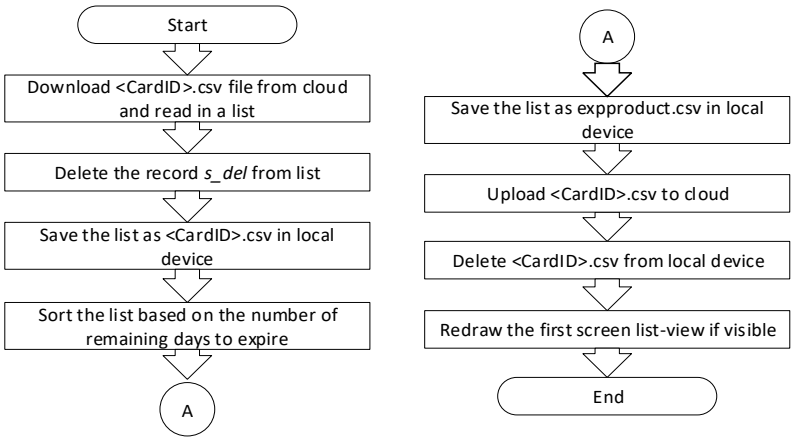


Figure 3. *Flowchart of deleting an item*

3.4. IoT Device

In the proposed smart expiry system, an Internet of Things (IoT) device that can be attached to a refrigerator or on a wall - shows the list of expiring items, current time, and weather gadget. The device also read out loud the soon-expiring items once in a day whenever it detects a motion nearby. The hardware and software parts of the device are described below.

3.4.1. Hardware. The block diagram of the hardware unit is shown in Fig. 6. A Raspberry Pi (RPi) v3 [17] single board computer is used as the main processing unit. It containing a 1.2 GHz 64-bit quad-core ARMv8 microprocessor, 1 GB of RAM, micro SD card slot supporting up to 32 GB, 40 general purpose input output (GPIO) pins, onboard Wi-Fi module, and other built-in hardware peripherals such as universal asynchronous receiver/transmitter (UART), I2C, serial peripheral interface (SPI), universal serial bus (USB), audio output via 4-pole 3.5mm connector, LCD interface (DSI) etc. A 7-inch capacitive touch LCD [18] is interfaced with the RPi using the DSI and the I2C port. The LCD has 24-bit colour depth and the screen resolution of 800 × 480 pixels. A passive infrared (PIR) motion sensor [19] is connected with a GPIO pin of the RPi board for detecting nearby motion. The alarm pin of the sensor goes to low voltage whenever a motion is detected. As the alarm pin is an open collector, a 1 kΩ pull up resistor was connected between the alarm pin and the 5v power supply. A personal computer (PC) speaker is connected to the RPi board using the 3.5mm connector to generate the text-to-speech sounds. Finally, the power supplies for the RPi board and the touch LCD are supplied using 110V AC to 5.1V DC adapters [20].

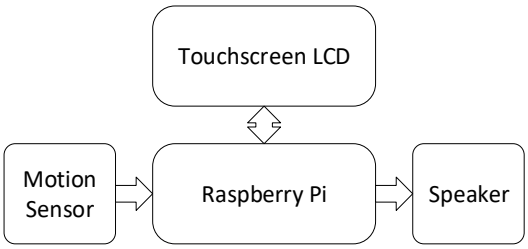


Figure 4. Block diagram of the hardware unit of the IoT device

3.4.2. App. The Android Things [21] embedded operating system is installed in the RPi. Android Things is recently developed by Google, aimed to be used with low-power and memory constrained IoT devices. The platform can run any program written targeting Android and it also comes with libraries for accessing hardware peripherals such as UART, I2C, and GPIO of the RPi. The device gets connected to the home Wi-Fi and can access the Internet.

The smart-expiry app for this device is written having the similar structure of the phone app with some additional features. The first screen of the app contains a list-view box. Each record in the list-view box shows the name of the item, days remaining to expire or how many days ago it expired, and the actual expiry date. It also shows the current time and weather widget on the screen. Hypertext Markup Language (HTML) code was embedded in web views of the app for fetching time [22] and weather [23] information from the Internet.

The app contains a settings menu for configuring the smart expiry card ID, synchronization interval with the cloud, and voice notification. In order to use the IoT device, the user must enter the card ID in the settings. From synchronization interval with cloud settings, the user can specify the interval in minutes, *SyncInterval*, after which data is downloaded from the cloud periodically. A service as shown in Fig. 2(a) is used to download the files from the cloud.

From the voice notification settings, the user can choose whether voice notification will be activated when nearby motion is detected, days before voice notification, *DaysBeforeVoiceNotification*, and voice notification hour of the day, *VoiceNotificationHour*. The app detects any change to the GPIO pin where the alarm pin of the motion sensor is connected. When motion is detected - the pin changes from high to low and the app reads out loud the items that are expiring within *DaysBeforeVoiceNotification* days or already expired. A text to speech library is used to convert the string to voice. It then sets a flag, *isSpeakToday*, so that the voice notification is not activated until next day when motion is detected. A service, *ServiceSpeakHourReset*, starts every day at *VoiceNotificationHour* and it resets the *isSpeakToday* flag. Whenever a food is consumed, it can be removed from the list by long-clicking the item similarly as discussed in Section 3.3.4.

IV. RESULTS

A prototype of the proposed smart expiry system as shown in Fig. 1 is developed and tested successfully for different cases. In the proposed system, the customer possesses a smart-expiry magnetic card as shown in Fig. 5. Each card has a unique card ID and the ID is written in the magnetic strip using a card reader/writer [26]. The card ID is also written on the top the card using English letters and also in scannable QR code.

(a) (b)

Figure 5. Smart-expiry magnetic card: (a) Top view - containing the card ID in QR code and in English letters; (b) Bottom view - containing the card ID written in the magnetic strip.

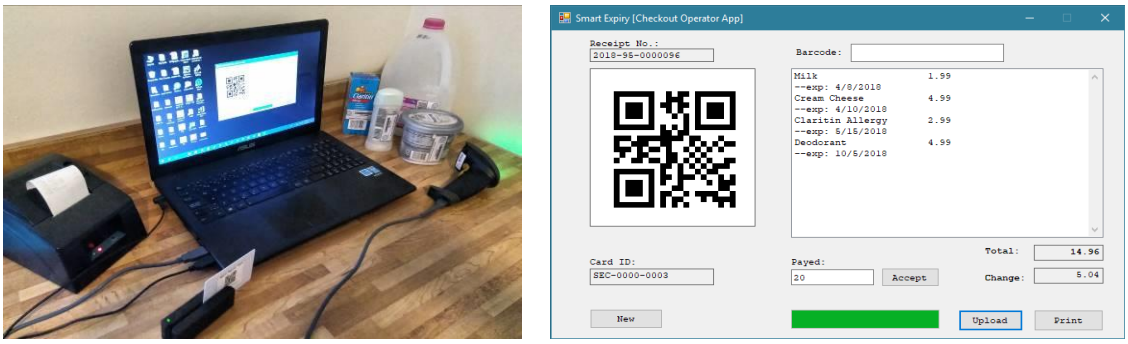
In the proposed system, the expiry date is written on the product using scannable barcode along with English as shown in Fig. 6. The desktop computer app for the checkout operator, as discussed in Section 3.1, is developed for Windows platform. The computer setup and a screenshot of the checkout operator app are shown in Fig. 7.

(a) (b) (c)

Figure 6. The expiry date is written using barcode along with English: (a) on medicine product; (b) food product; (c) cosmetic product

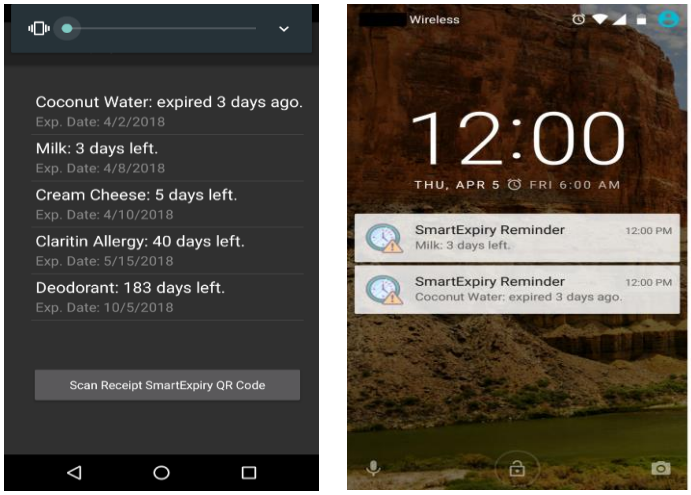
The barcode for the product and also the barcode for the expiry date is scanned by the checkout operator using a barcode reader as shown in Fig. 7. This step is repeated for all the purchased products which have expiry dates. The app decodes the barcodes, gets the product names from the store's barcode to product name conversion database, and creates a new table which contains the purchased product names and expiry dates.

In this proposed method, if the customer gives a smart-expiry card to the checkout operator for swiping, then the app uploads the table file (having the same name of the card ID) to the Firebase storage cloud, as discussed in Section 3.2, using the Internet. This table is downloaded automatically by the smartphone app of the customer, as discussed in Section 3.3.2 (Case 1), without any human intervention.



(a) (b)

Figure 7. (a) Checkout operator computer with a barcode scanner, card reader, and a receipt printer; (b) Screenshot of checkout operator app;



(a) (b)

Figure 8. Screenshot of the smartphone app (a) the first-screen showing the list of items that are going to expire or already expired; (b) Notification generated automatically at a specific hour of the day for the soon-expiring items.

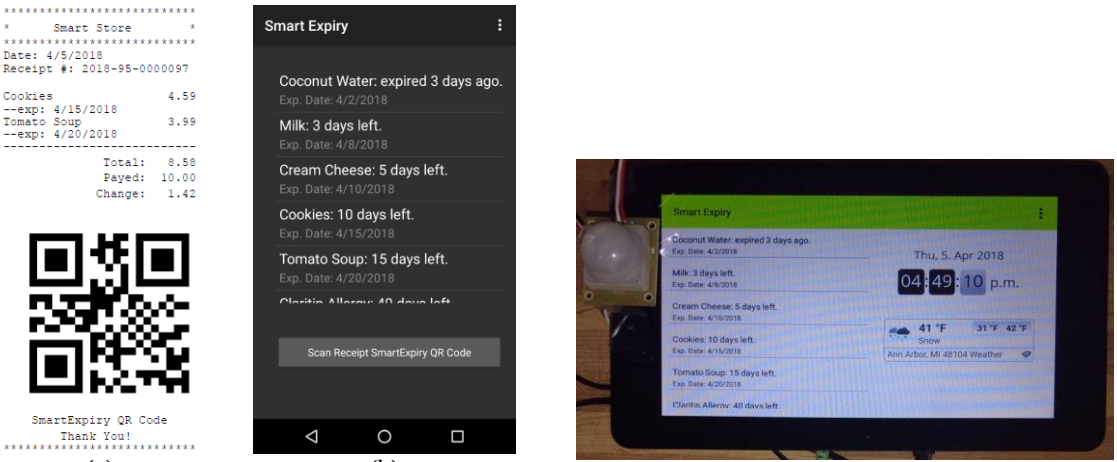
The smartphone app is developed according to the discussion in Section 3.3. A screenshot of the first screen of the smartphone app is shown in Fig. 8(a). Here, each row of the list-view box shows the name of the item, days remaining to expire or how many days ago it expired, and the actual expiry date. The app generates notifications for the soon-expiring items by calculating the date difference between the expiry date and the current date. A screenshot of notifications is shown in Fig. 8(b).

The IoT device, as shown in Fig. 9, is developed as discussed in Section 3.4. This device automatically downloads the table from the cloud. It shows the list of expiring items, current time, and weather gadget. The device also read out loud the soon-expiring items once in a day whenever it detects a motion nearby.

(a) (b)

Figure 9. The IoT device: (a) Top view – showing a list of expiring items, current time, and weather gadget. The motion sensor is attached to top-left of the screen (b) Bottom view –Raspberry Pi board interfaced with LCD and motion sensor.

In the proposed system, if the customer does not give a smart-expiry card to the checkout operator for swiping, then the app uploads the table file (having the same name of the receipt number) to the Firebase storage cloud using the Internet. After the payment is done, a receipt is printed, as shown in Fig. 10(a), having conventional product names and prices along with a QR code representing the receipt number. The QR code in the receipt is scanned by the smartphone app by the customer and the expiry dates with product names are downloaded from the cloud to the phone, as discussed in Section 3.3.2 (Case 2), and also in the IoT device as shown in Fig 10(b) and Fig. 10(c) namely.



(a) (b) (c)

Figure 10. (a) when the smart-expiry card is not used, QR code is printed on receipt; (b) the QR code is scanned by the customer's smartphone and the list is updated. (c) updated list of the IoT device.

A comparison of the proposed work with other related works is shown in Table 1. Compared with these related works, the proposed cloud-based architecture seeks to remedy the tiresome manual entry of each product information in the smartphone. Moreover, it provides notification in the smartphone and on an IoT device that can be attached to the refrigerator on any wall in the house. The IoT device shows graphical clock and weather gadgets. It also notifies the user using the voice of the expiring items once a day whenever it detects motion nearby.

Table 1. Comparison with other works

Work	Cloud Architecture	Automatic Data Entry	Notification in Smartphone	Notification in IoT device	Gadgets on IoT device
[11]	No	No	Yes	No	No
[12]	No	No, need to manually scan each product's barcode with a smartphone camera.	Yes	No	No
[13]	No	No, need to manually scan each product's name with a smartphone camera.	No	No	No
[16]	No	No, need to manually set expiry dates for each visible product in the refrigerator	No.	Yes, in the display device of the refrigerator.	Yes, in the display device of the refrigerator.
[10]	No	No, need to manually scan each product's QR code.	No	Yes, on the device.	No
Proposed	Yes. Data is synchronized among cloud, smartphone, and the IoT device.	Yes, no human intervention when smart expiry card is used. Needs to scan only a single QR code when no card is used.	Yes, with sound and vibration	Yes, text to speech technology to read out loud the expiring items when motion is detected nearby.	Yes, displays digital clock and weather gadget.

V. CONCLUSION

In this paper, a cloud-based smart expiry system is proposed. In the proposed system, expiry data is automatically downloaded to the customer's smartphone – without the hassle of manual entry. The system generates notifications of the soon-expiring items on the smartphone and on the IoT device. A prototype of the system has been developed and tested. Future work includes interfacing the system with Amazon Echo and developing the smartphone app on iOS platform.

ACKNOWLEDGMENT

The author would like to thank Mohammed Rashed Hyder for his technical support of the cloud. This work is supported by James H. Brickley Endowment for Faculty Award of Eastern Michigan University.

REFERENCES

[1] E. B. Leib, C. Rice, R. Neff, M. Spiker, A. Schklair, and S. Greenberg, "Consumer perceptions of date labels: national survey," *Food Waste Summit*, May 2016. [Online]. Available: http://www.chlpi.org/wp-content/uploads/2013/12/Consumer-Perceptions-on-Date-Labels_May-2016.pdf
[2] Waste in America: Statistics and facts on food & water, *Sound Vision*, 2018. [Online]. Available: http://www.soundvision.com/article/waste-in-america-statistics-and-facts-on-food-water
[3] T. W. Jones, "Using contemporary archaeology and applied anthropology to understand food loss in the American food system," *Report by Bureau of Applied Research in Anthropology*, University of Arizona, 2004.
[4] K. D. Hall, J. Guo, M. Dore, and C. C. Chow, "The progressive increase of food waste in America and its environmental impact," *PloS one*, vol. 4, issue. 11, 2009. doi: http://dx.doi.org/10.1371/journal.pone.0007940.
[5] P. Mitchell, "Health effects of eating expired foods," 2015, [Online]. Available: http://www.livestrong.com/article/485198-health-effects-of-eating-expired-foods/
[6] Molds on Food: Are They Dangerous?, *United States Department of Agriculture Food Safety and Inspection Service*, 2013. [Online]. Available: http://www.fsis.usda.gov/wps/wcm/connect/a87cdc2c-6ddd-49f0-bd1f-393086742e68/Molds_on_Food.pdf?MOD=AJPERES
[7] QR code standard (ISO/IEC 18004), 2000, [Online]. Available: http://www.swisseduc.ch/informatik/theoretische_informatik/qr_codes/docs/qr_standard.pdf
[8] P. Gharjale and P. Mohod, "Efficient public key cryptosystem for scalable data sharing in Cloud storage," *Computation of Power, Energy Information and Commuincation (ICCPEIC)*, Chennai, pp. 0325-0329, 2015.
[9] M. B. Rashid, N. Islam, A. A. M. Sabuj, S. Waheed and M. B. A. Miah, "Randomly encrypted key generation algorithm against side channel attack in cloud computing," *Electrical Engineering and Information Communication Technology (ICEEICT)*, Dhaka, pp. 1-5, 2015.
[10] R. Darell, "QR fridge magnets keep track of your food's expiration dates," 2013, [Online]. Available: http://www.bitrebels.com/technology/qr-fridge-magnets-food-expiration/
[11] Best Before Manager app, 2018, [Online]. Available: https://play.google.com/store/apps/details?id=com.bokang.myproj.bestbeforemanagerfree
[12] Best Before TPP app, 2018, [Online]. Available: https://play.google.com/store/apps/details?id=pl.jacek.jablonka.android.tpp
[13] E. Peng, P. Peursum, and L. Li, "Product barcode and expiry date detection for the visually impaired using a smartphone," *in Proc. Digital Image Computing Techniques and Applications (DICTA)*, pp. 1-7, Fremantle, WA, 2012.
[14] Firebase Storage, 2018, [Online]. Available: https://firebase.google.com/docs/storage/
[15] Android Service, 2018, [Online]. Available: Ihttps://developer.android.com/guide/components/services.html.
[16] Samsung Family Hub refrigerator, 2018, [Online]. Available: http://www.samsung.com/us/explore/family-hub-refrigerator.
[17] Raspberry Pi, [Online]. Available: https://www.raspberrypi.org, 2018.
[18] Raspberry Pi LCD - 7" Touchscreen, [Online]. Available: https://www.sparkfun.com/products/13733, 2018.
[19] PIR Motion Sensor, [Online]. Available: https://www.sparkfun.com/products/13285, 2018.
[20] DC Power Supply, [Online]. Available: https://www.sparkfun.com/products/13831, 2018.
[21] Android Things, [Online]. Available: https://developer.android.com/things/get-started/index.html, 2018.
[22] Clock widget, [Online]. Available: https://www.zeitverschiebung.net/en/clock-widget, 2018.
[23] Weather widget, [Online]. Available: https://www.willyweather.com/widget/create.html, 2018.
[24] DateTime.Ticks Property, [Online]. Available: https://msdn.microsoft.com/en- us/library/system.datetime.ticks(v=vs.110).aspx, 2018.
[25] Primary Key [Online]. Available: https://www.techopedia.com/definition/5547/primary-key, 2018.
[26] MSR605X Magnetic Stripe Reader Writer, [Online]. Available: https://www.walmart.com/ip/New-MSR605X-HiCo-Magnetic-Card-Reader-Writer-Encoder-MSR605-MSR206-MSR606/191791357, 2018.

