iShop: A Continuously Context-Aware Shopping Assistant

Alaa Abdulaal
Dept. of Computer Science
University of Toronto
Toronto, Canada
alaa.abdulaal@mail.utoronto.ca

Jacqueline Bermudez
Dept. of Computer Science
University of Toronto
Toronto, Canada
jbermudez@cs.utoronto.ca

Jyotheeswar Arvind Manickavasagar
Dept. of Computer Science
University of Toronto
Toronto, Canada
jyotheeswar@cs.toronto.edu

ABSTRACT

We present iShop, a continuously context-aware shopping assistant mobile application that keeps track of the user's location, reminding them of the time spent at the store and presenting the store membership cards. iShop has a user-friendly interface allowing users to search for a store location where the user would like to set a shopping timer. Users can add timers, edit, and delete them. Also, users can scan their loyalty or membership cards using the camera and add them with the option of editing and deleting those cards at any time. Two levels of evaluation were done with iShop: a user study and a technical evaluation, with particular concentration on the battery and network bandwidth consumption. The user study results showed that 50% of users would like to use iShop. Also, the technical evaluation results showed that battery consumption was reduced by lazily pre-populating geo-fences.

Categories and Subject Descriptors

C.5.3 [Microcomputers]: Portable devices; K.8 [Personal Computing]: General

General Terms

Performance, Design, Reliability, Human Factors.

Keywords

Continuous tracking, context aware, digital wallet, location-based services, GPS, Wi-Fi, time tracking.

1. INTRODUCTION

Current advancements in technology allows software engineers and developers to build customizable applications (apps) to address user needs. One such example, is the development of apps designed to run on mobile devices such as smartphones and tablets. Mobile apps have contributed in many areas such as health, education, entertainment, business and other fields. It is worth mentioning that some of the best mobile apps that have succeeded, use the Global Position System (GPS) navigation systems' data. The most common apps that include GPS functionality are generally related to maps, calendars, social networks and trackers.

With the rat race of everyday life, people have become ever busier and filled with activities. Thus, it is common for people to be overloaded with information, leading them to forget important but mundane tasks that they have to do. Moreover, one of the most common problems that people normally encounter is losing track of time in supermarkets and shopping stores. Attempts have been made to solve this problem by using notes, calendars, timers and alarms. Although these solutions certainly help in some way, they can be futile and easily forgotten. Another activity that wastes user's time and is inconvenient is looking for membership and loyalty cards. Users either forget the card, or lose time searching for one in their wallet. New technology presents solutions to this problem, such as digital wallet apps when using smartphones but those application also make the user lose time by requiring the user to search through the digital cards.

Motivated by solving those problems, we introduce iShop, a shopping assistant mobile app that stores a digitized version of a membership card and tags it to related store locations. It also contextually identifies the location of a user, and pops up the corresponding card, if it is already stored, to avoid searching for a card manually. Finally, it keeps track of time spent at a store, and sends a reminder if users have spent more than the planned duration.

The rest of this paper explores this solution in detail. Section 2 describes the overall architecture of iShop. Section 3 delves into how iShop has been implemented. The core of iShop, the tracking and predictor, is described in Section 4.

2. iSHOP SOLUTION

2.1 System Overview

The solution has five main hardware/software components:

2.1.1 GPS Receiver

Almost all modern smartphones have a built-in GPS receiver that can receive radio signals from GPS satellites. Using the readings from both the GPS and the mobile network/Wi-Fi Internet reverse geocoding, the application can estimate the user's location.

2.1.2 Mobile Network/Wi-Fi internet connection

Smartphones can connect to the internet using either mobile networks or Wi-Fi access points. The application connects to the Google servers using the Google Maps API v2.0 [4] over the internet, and does the geocoding needed to get the map, store name, and location information.

2.1.3 Camera

All modern smartphones have a built-in camera making it convenient for card scanning and digitization. The camera scan the barcode from the card recognizing the white and black stripes and generating the matching barcode number.

2.1.4 Database
The database (DB) is designed to store the timers saved by the user, cards saved by the users, and locations saved by the application.

2.1.5 Application & UI
The application user interface (UI) is designed to be user friendly. The main page provides a direct access to the application's main functionalities. The "Add timer" is based on a google map interface and a search box to find locations where the user would like to add a timer. A sliding panel that pops on top of the map is used to add the timer details easily, without the need to open another page. The "My Timer" enables the users to view the list of un-reminded timers, edit, or delete them. The "Add card" is based on a simple camera interface to scan the card barcode then enter the card store name. Users also can easily edit and delete existing cards from the "My cards" list.

2.2 System Architecture
Figure 1 shows our solution's architecture, demonstrating all the essential components and functions which will be discussed in details in section 3.

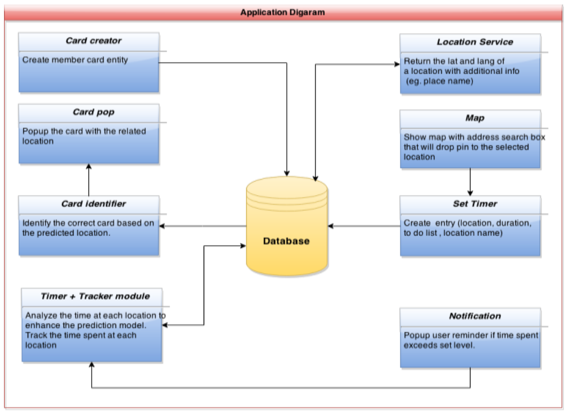
Application Digaram
Card creator
Create member card entity
Card pop
Popup the card with the related location
Card identifier
Identify the correct card based on the predicted location.
Timer + Tracker module
Analyze the time at each location to enhance the prediction model. Track the time spent at each location
Database
Location Service
Return the lat and lang of a location with additional info (eg. place name)
Map
Show map with address search box that will drop pin to the selected location
Set Timer
Create entry (location, duration, to do list , location name)
Notification
Popup user reminder if time spent exceeds set level.

Figure 1. iShop solution architecture

3. iSHOP IMPLEMENTATION
The application was developed on the Android Studio development environment. During the development stage, the application was debugged and tested on three devices, each running Android 5.0: a LG Nexus 5, a Galaxy Core Lite, HTC 1M8, and Motorola G (1st Generation).

3.1 Timer
3.1.1 Add Timer
The user sets the specific time they would like to spend at a specific location by selecting "New Timer" button. If the user chooses to save a new timer, the application will save the information in the DB. After creating the timer, the app will be included in a cycle of verification where each 10 seconds the app verifies if the user is at the location that was set. Otherwise, the timer won't start until the user arrives to the specified location. The timer starts once the user arrives at the correct location, and the user will receive a notification when the countdown timer finishes.

Map environment
The application uses the Google Maps Android API v2[4]. The location can be placed on the map by typing the name of the establishment. In order to make the map work on the smartphone, it must be connect to Wi-Fi, or to a cellular data service.

Store name Input
While the user types in the search box from the map, autocomplete address suggestions open in a list with a maximum number of 5 suggested addresses. Once an address is selected from the list, a pin is dropped on the geocoded geographic latitude and longitude of the selected address. This autocomplete feature uses the Google Places API [5] that provides detailed information about places across a wide range of categories, in this case establishments, from the DB of Google Maps and Google+. Establishments in this study were from Ontario, Canada; however, it could be modified for other places outside Canada. One of the limitations of using this autocomplete feature, is that the free version only allows 1,000 queries per 24 hours, and the establishments returned are limited.

3.1.2 Manage timer
A user interface is available, showing the list of timers created. User are able to modify the timer and the description of things that the user wants to do at a specific place. The user is able to delete the chosen created timer if he desires. This information will be either updated or deleted from the local DB according to the user's preferences.

3.2 Cards
3.2.1 Add cards
A new entity is created in the DB when a new card's barcode is scanned and saved by the user. The application uses an open-source library, called Zxing ("zebra crossing"). Zxing is multi-format 1D/2D barcode image processing library implemented in Java, with ports to other languages. The supported formats are UPC-A, Code 39, QR Code, UPC-E, Code 93, Data Matrix, EAN-8, Code 128, Aztec (beta), EAN-13, Codabar, PDF 417 (beta), ITF, RSS-14, and RSS-Expanded [9].

3.2.2 Manage cards
The user is able to modify the cards either by deleting them it or updating the name of the store. Based on the card ID and the modification, the application will check the card table in the DB for the matching card ID and update it. At the same time, the application will check the Address table in the DB to delete/ update the corresponding address entry related to that card.

3.3 Application Database
The DB uses the SQLite framework standardized across Android devices, and as such requires minimal setup and administration of the DB. The DB has three main tables.

3.3.1 Timer table
The timer table saves the users' added timers with the attributes: Timer ID, Timer to-do list, isReminded - a flag to

indicate if this timer has been reminded already, latitude and longitude of the selected location, and the timer duration the users specify. Each attribute has an API that interacts with it. The application will constantly read the location of the timers stored in the DB and compares it with the user's current location, to figure out which timers need to be triggered.

3.3.2 Cards table

The cards table saves the users added cards with the attributes: cards ID, store name that the card will be used in, and the generated barcode number from the scanned card. Each attribute has an API that interacts with it.

3.3.3 Address table

The address table saves the user added cards address with the following: store ID, store name that the card will be used in, store location latitude and longitude.

In the initial version of iShop, this table was not present. The application, at every 10 second time slot, uses the user's current location and makes a Google Maps v2.0. Nearby Places API call to obtain the list of locations that are near to the user. The 10 second time slot was chosen so that there was sufficient time for the application to make the API call, get the results and parse it before being refreshed. However, in our initial testing, we found this to be a highly inefficient solution, which led to major battery and bandwidth consumption.

Optimization

In order to ameliorate this issue, we decided to lazily pre-populate the latitudes and longitudes of all outlets of a particular shop (e.g. if a user creates a membership card for Sobeys, the application would immediately find the locations of all Sobeys' stores nearby), make a permanent geo-fence, unlike Google's geo-fences, and stores all the information into the Address Table.

The application uses these locations mapped with a particular card, and compares it with the user's current location in order to perform its functions. When the pop-up is generated, the application maps this information to that in the Cards table in order to generate the particular membership card for the user.

3.4 Location service

The Location service was implemented using Android's Location & Places API. The application obtained the user's location using GPS data, and refreshed this every time the user moves. This service also holds a cache so that the last location is used by the application, in the case that GPS is not available (e.g. while shopping indoors at malls).

3.5 Card Prediction & Time Calculation

The card prediction and Time calculation modules are predominantly responsible for identifying and populating the correct membership card for the user. The card prediction module would populate the membership card upon the set time being elapsed for the user, whereas if the user doesn't have a timer, but goes to a store where he/she has a card, the application attempts to predict which store they are at. iShop does this by initially checking that the user is not in the vicinity of a timer (as timers have a higher priority because of user-defined relations). If there are no timers, the application predicts by comparing the user's locations with the geo-fences stored in the Address table to see if the user is close to any of them.

When iShop identifies that the user is within a geo-fenced region, it immediately starts timing the user. We have set a threshold of 3 minutes for the prediction with no-reminder case, as we found that any lesser time would lead to large false positives. These false positives are triggered because the user may be traveling (walking/biking/in public transit) over a geo-fenced region without meaning to enter the store. They may have stopped there due to traffic or red signs. A low threshold would lead to the application repeatedly identifying that the user is shopping at location, when he/she is not. Repeated false reminders would make the application irritable for the user.

3.6 Notification

The notification based on the information obtained from the card predictor, the Notification module pushes a notification/reminder to the user that they have a membership card available at a particular location. When the user clicks this card, the list of predicted cards is populated and the user can click on the appropriate card to display it. There are two types to this module: (i) Timer notification and (ii) Predictor notification.

3.6.1 Timer notification

By design, the application considers timers to have higher priority. If the notification is of a timer, then the application would only pop-up the notification stating that the time the user has set to shop has elapsed. On clicking the notification, the user can access their membership card.

3.6.2 Predictor notification

The application tries to identify the user's precise location. This is non-trivial, as there are many stores that exist next to each other and the user may have membership cards in two or more of them. With the limited granularity of GPS locations, the application may assume the user is in shop X, whereas the user would actually be in shop Y next door. iShop attempts to do this by fitting a linear model to user shopping behavior. Whenever the user chooses a card, the information is stored into the timer table in the DB with `status=reminded`. Based on the user's previous shopping history, the application first identifies all the locations nearby (radius of 125m) where the user has a membership card. Based on user history, the application sorts these locations and stores them in the order of maximum probability of shopping. This calculation is analogous to the Support and Confidence values used by the Apriori Algorithm [1].

4. iSHOP TIMER & TRACKER MODULE

The iShop engine consists of a Location service and a background service that serves as brain of the application. Figure 2 shows the engine's functionality.

Algorithm:

```
ishop_engine(){
    Location l=getCurrentLocation();
    for(i=0 to i<reminder_table.size())//for reminder
    if(distance_from(reminder_table.get(i).location,l)<150)&&timer_not_started)
    start_timer;
    else if(moved_away)
      stop_timer;
    for(j=0 to j<address_table.size()) //for prediction
    if(distance_from(address_table.get(i).location,l)<150)&&timer_not_started)
    start_timer;
    else                          stop_timer;
    if(timer_complete)
      if(reminder)
        pop_up_reminder_card;
      else //prediction
        for(i=0 to j<cards_table.size())
    if(distance_from(cards.(i),l)<150)
    add_data(list_cards)
    filter(list_cards) // linear model to only show cards that have been frequently used
    sort(list_cards,DESC); }
```

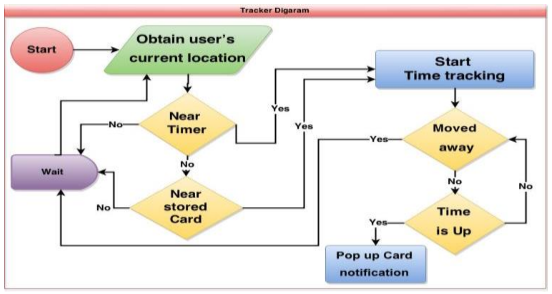

Figure 2. iShop engine

The Google Places API has a restriction of only returning 20 locations at a particular instance. iShop deals with this by obtaining the location when a card is created. When the user moves out of the boundary from which the initial 20 locations have been obtained, iShop makes another API call in-order to expand its list of locations. The drawback of this approach would be that if the user deviates from his/her usual route in the beginning, the list of locations may grow to be very large. iShop attempts to alleviate this problem by dropping stale location rows. Although this may cause the application to show non-ideal behavior in certain situations, we believe it to be a fair trade-off. Further evaluation is needed to see if this approach would work in a wider setting.

5. EVALUATION

Our application has been evaluated in two parts: (i) User study and (ii) Technical evaluation, with concentration on the battery and network bandwidth consumption.

5.1 User study

10 participants between the ages of 22-61 were recruited from Toronto, ON and Jordan, ON Canada. They were drawn from a variety of professions to ensure a wide audience for testing the application. We performed a qualitative study on the application, with questions targeted to get feedback on the usability of the application, on how accurate the prediction worked, on how accurate the reminders worked, and the accuracy of the timers.

As mentioned in [10], the qualitative study provided us with the information we required. The application was installed on the participants' mobiles, running Android, and were given an initial demo of the application. Users were then allowed to test the application over two days, while continuing with their daily routine. After testing, participants filled an online form with their responses. Figure 3 graphically represents the user's feedback.

With regards to how many users would use our application, we received a positive feedback of about 50% of users stating they would like to use our application, albeit with a few of them requesting additional functionalities. Some of the free-text constructive feedback received from users included: "It would be useful to track indoor shops", "include track of the balance of each card after the customer uses", "include store profiled, shopping list and hours".

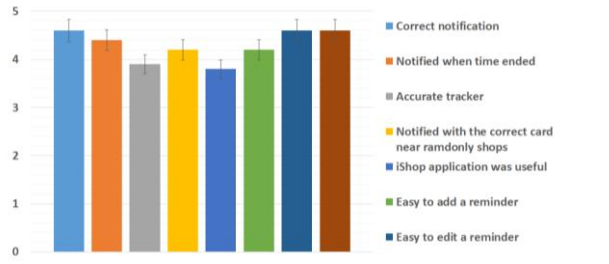

Figure 3. User Survey Results

5.2 Technical evaluation

The application was evaluated on the battery and network bandwidth consumed by the application, and the radius threshold for the geo-fencing of the application. The application was evaluated on a Nexus 5 smartphone (Android 5.0, Qualcomm Snapdragon 800 processor, Wi-Fi 802.11 a/b/g/n/ac, dual-band) using Wi-Fi and a Motorola G 1st Generation (Android 5.0, Qualcomm Snapdragon 400 processor) using 3G network.

5.2.1 Battery Consumption

The battery consumption was tested using Power Tutor 1.4. iShop was run on Wi-Fi using the Nexus 5 and over the 3G network on the Moto G. It was run for approximately 5 hours and 7 hours on the phones respectively, storing 3 cards, setting 2 reminders and walking into a shop where the user has a card stored. The battery consumption has been normalized for 12 hours and are shown in Figures 4 and 5.

The application power consumption was compared with the optimization and without optimization. The nearest application that had similar functionality was Google Maps, which showed power consumption similar to the optimized application while running on the foreground. Without optimization, iShop consumed ~34.8% of the phone's battery while over 3G. With optimization, the power consumption was ~3.6%.

5.2.2 Network Bandwidth consumption

The application bandwidth consumption was evaluated using ONAVO COUNT 2.2.2-3. This evaluation was done with the Moto G phone over the 3G network. As shown in Figure 6, with optimization, the application only consumed about 2MB per day whereas without optimization, it was around 33MB. Google Maps, while running on the foreground, consumed about 12MB per day. We attribute the reduction of bandwidth consumption due to the fact that pre-populating at periodic intervals drastically reduces the API calls and only requires CPU processing with the GPS data.

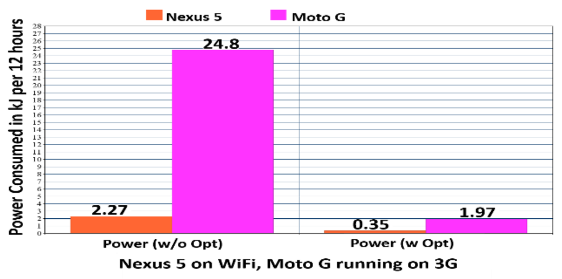

Figure 4. The battery consumption has been normalized for 12 hours

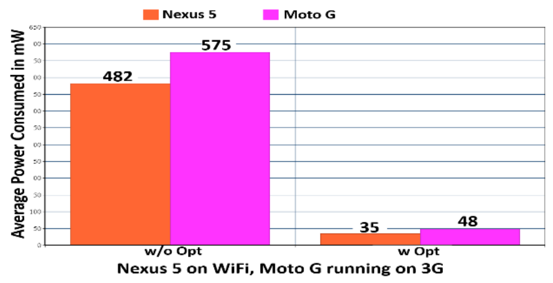

Figure 5. Battery consumption

5.2.3 Radii threshold

For the scope of our application, it is vital for the geo-fencing to be of sufficient radius so that the application can detect an appropriate number of stores for it to predict properly.

However, this involves making a trade-off. Having a small radius would make the application detect only a limited number of useful stores, possibly leading to false negatives, whereas a large radius would lead to false positives and a lot of noise. To this effect, we evaluated on the ideal threshold for our application. This was done by varying the radius and walking around different localities in downtown Toronto to identify which radius provided a fair balance between the number of shops identified and the noise. Figure 7 shows the chart obtained by plotting this information.

6. RELATED WORK

Location-based task reminders function when users are able to establish personal-meaningful locations and create location-based task reminders. Given the desired information, the application triggers the reminder when user is at the predefined location [7]. Another type of apps that are finding increasing usage among users are digital wallets [2, 3]. These apps allows the user to digitally store loyalty/membership card data [6], electronic cash and receipts. This is a good solution to avoid filling user's wallets with cards but user still lose time searching through the stored digital cards. Both type of apps, digital wallet and location-based reminders, are combined in our solution to make life easier for people when shopping in stores. It hasn't been found any official research study that introduce this combined idea.

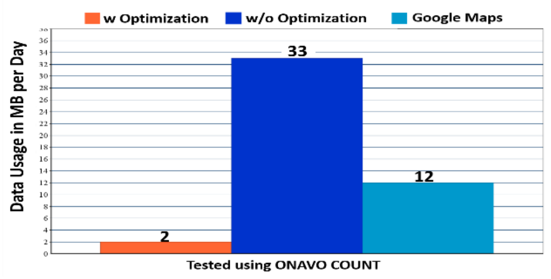

Figure 6. Network Bandwidth consumption

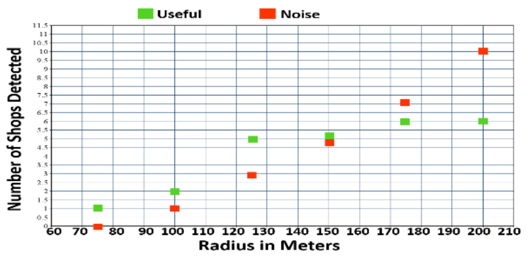

Figure 7. # of shops returned with different radius radii Toronto

7. FUTURE WORK

With limited number of shops in a vicinity, the linear prediction model used by our application is sufficient. However, when the application is extended to include indoor location information, particularly in the scenario of shopping in a shopping mall, this would be less than ideal taking into account the large granularity in obtaining the user's position. Future work should be focused on improving this prediction model to better fit the user's data. In addition, currently in the application the radius used to check the user's location against the store name is fixed (125 meters) for all stores. A future enhancement would be working on making the application modify the radius threshold dynamically based on the type of establishment and using architecture information from TSA databases. This has been shown to better user-experience with similar apps [8]. Finally, the application can be enhanced by taking advantage of the cloud, pushing the data to a web server and analyzing it.

8. CONCLUSION

We have thus created iShop, a continuously context-aware shopping assistant that digitizes loyalty or membership cards and predicts user shopping behavior. The initial version of iShop involved obtaining a list of nearby locations every 10 seconds and trying to match it with the list of reminders or membership cards. However, this was highly inefficient and thus the application was re-engineered to optimize this process by lazily pre-populating the list of stores. We show that this optimization leads to a tremendous decrease in battery and bandwidth consumption, while the CPU usage difference is negligible. With our user study, we find that there is a potential market for this application.

9. REFERENCES

[1] Agrawal, R., Imielinski, T., and Swami, A. Mining association rules between sets of items in large databases. *ACM SIGMOD*, 1993.
[2] Bradley, Michael. Digital Wallets Executive Briefing. *Information Technology Association of Canada (ITAC) Digital Commerce Forum: How Your Wallet is Going Digital*, 2013.
[3] Crowe, M., and Tavilla, E. (2012). Mobile Phone Technology: "Smarter" Than We Thought. *Federal Reserve Bank of Boston*, 136-149.
[4] Google developers. 2015. Google Maps API. Google Maps Android API v2. https://developers.google.com/maps/documentation/android/
[5] Google developers. 2015. Google Places API. Google Places Android API v2. https://developers.google.com/places/
[6] Hopken, W., Deubele, P., Holl, G., Kuppe J., Schorpp, D., Licones, R., and Fuchs, M. Digitalizing Loyalty Cards in Tourism. *Information and Communication Technologies in Tourism*, 2012.
[7] Lin, C. and Hung, M. A location-based personal task reminder for mobile users. *Personal Ubiquitous Comput*. 18, (2), 2014, 303-314. http://dx.doi.org/10.1007/s00779-013-0646-2.
[8] Ludford J. P, Frankowski, D., Rilyey, K., Wilms, K., and Terveen, L. Because I Carry My Cell Phone Anyway: Functional Location-Based Reminder Applications. *ACM SIGCHI* 2006, Montreal.
[9] Owen, S. ZXing—Multi-format 1D/2D barcode image processing library with clients for Android, Java, 2009.
[10] Ravasio, P., Guttormsen-Schar, S., and Tscherter, V. The Qualitative Experiment in HCI: Definition, Occurances, Value and Use. *Transactions on Computer-Human Interaction*, 2004,1-24. http://pamela.shirahime.ch/QualExp.pdf

.