Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE

A Privacy-Preserving Mobile LBS System for Small Businesses

Ahmed Abdelmoamen Ahmed*

[1]Department of Computer Science Prairie, Prairie View A&M University, TX, USA

Correspondence
*Corresponding Ahmed Abdelmoamen Ahmed. Email: amahmed@pvamu.edu

Summary

Small-to-medium businesses are always seeking affordable ways to advertise their products and services securely. With the emergence of mobile technology, it is possible than ever to implement innovative Location-based Advertising (LBS) systems using smartphones that preserve the privacy of mobile users. In this paper, we present a prototype implementation of such systems by developing a distributed privacy-preserving system, which has parts executing on smartphones as a mobile app, as well as a web-based application hosted on the cloud. The mobile app leverages Google Maps libraries to enhance the user experience in using the app. Mobile users can use the app to commute to their daily destinations while viewing relevant ads such as job openings in their neighborhood, discounts on favorite meals, etc. We developed a client-server privacy architecture that anonymizes the mobile user trajectories using a bounded perturbation strategy. A multi-modal sensing approach is proposed for modeling the context switching of the developed LBS system, which we represent as a Finite State Machine (FSM) model. The multi-modal sensing approach can reduce the power consumed by mobile devices by automatically detecting sensing mode changes to avoid unnecessary sensing. The developed LBS system is organized into two parts: the business side and the user side. First, the business side allows business owners to create new ads by providing the ad details, Geo-location, photos, and any other instructions. Second, the user side allows mobile users to navigate through the map to see ads while walking, driving, bicycling, or quietly sitting in their offices. Experimental results are presented to demonstrate the scalability and performance of the mobile side. Our experimental evaluation demonstrates that the mobile app incurs low processing overhead and consequently has a small energy footprint.

KEYWORDS:
Location-based; Privacy; LBS; Mobile; Advertising; Android

1 | INTRODUCTION

In the era of mobile and smart technologies, it is becoming increasingly affordable to use Location-based Advertising (LBS)[1,2] by small-to-medium businesses for advertising products and services. LBS is a location-based service that uses real-time geographical information collected from smartphones to create effective campaigns. LBS could also be used to recruit temporal or permanent employees in a small business hosted in a local neighborhood. LBS enables business owners to personalize their messages to customers based on their current location in real-time. Using a customer's geographical coordinates, advertisers can

send different messages to people depending on their current geo-location. This has removed the barriers between marketplaces and customers by reaching to shoppers when they are proximate to advertisers' sites[3].

Imagine an advertisement system that can be easily used by business owners to create free ads for a job opening, discount on a newly offered meal, or brand new service, while potential customers in their neighborhood could easily view these ads conveniently using their smartphones. This system would create a better opportunity for business owners to target the right customers at the right time. Such systems can also be used by business owners to track the movement of customers passing by their stores, determine rush shopping hours, and create efficient staffing plans[4].

This paper presents a prototype implementation of an LBS-based system that is handy for small-to-medium businesses to advertise various types of ads[1]. We developed a distributed system that is organized with parts executing on the mobile devices as well as a web-based application hosted on a cloud server. The user side is prototyped as a self-contained mobile application where users can view ads in a convenient way while driving, walking, or relaxing at home. The business side is developed as a web application that allows business owners to publish their ads using a user-friendly interface. These mobile ads are delivered to customers who are within a certain radius of the business location.

Mobile ads target a fixed area and send meaningful ads to customers entering that defined boundary within a specific time-frame. The business side provides a dashboard for business owners to create a new ad, edit an existing ad, view applications in response to their ads, and approve/reject requests. Once the newly created ad is approved, it will be published on the interactive map on the use side. The business side is developed using PHP, MySQL, HTML5, CSS, JavaScript, and JSON. The user side is developed using the Android Development Environment, coupled with Google Maps APIs and MySQL Database.

One of the major concerns of any LBS-based system is the privacy of mobile user trajectories. Most of the existing LBS approaches[5,6] collect user locations over time with user identities periodically. This could lead to different kinds of personal privacy breaches if a leak of the identified trajectories occurs. For instance, a trajectory leak can be used by malicious parties to launch inference and linking attacks by exposing the mobile user's interest in places and behaviors over time. The trajectory information can also be used for spam advertisements and targeted-individual assaults. To address these challenges, we developed a bounded perturbation technique to anonymize user's trajectories against any privacy breaches to address these challenges. The developed privacy mechanism prevents our LBS-based system from generating any visiting trajectories in real-time using the Global Location Set (GLS)[6] strategy.

As the number of mobile ads grows, so does their energy demand on mobile devices. An important opportunity for conserving energy for such ads lies in optimizing sensing for the evolving context that a device is in. Consider, for example, a context-aware mobile advertising app that monitors a device owner's location at a low frequency; however, if the user is approaching an interesting place, it switches the mode to begin monitoring several sensors at a high frequency. We call this *multi-modal context-aware sensing*. Using the proposed multi-modal approach, we can model the dynamicity of a such context-aware mobile system that adjust their own behavior according to the context changes.

This paper presents experimental results on the performance and energy costs of using the mobile app. To evaluate the scalability of the mobile app, we measured the resources required to host every single ad. Also, these results demonstrate the scalability, performance and energy efficiency of the multi-modal sensing approach.

The rest of the paper is organized as follows: Section 2 presents related work. Sections 4 and 3 present the design and prototype implementation of the advertising system, respectively. Section 5 experimentally establishes the performance cost of using the system. Finally, Section 6 summarizes the results of this work.

2 | RELATED WORK

There have been several compelling projects –both in academia and industry– involving location-based services in different domains such as real-time traffic information[7,8], rescue efforts[9], health[10,11], entertainment[12], mobile crowdsensing[13] and advertising[14,15,16]. This section focuses on existing work that enables location-based services on smartphones for advertising purposes.

By nature, location-based services running on smartphones provide real-time information about its owner locations automatically. This information could be used by marketers to offer timely personalized services and products that are location-specific[4]. It is becoming feasible than ever to use real-time location information to link the existing knowledge of the consumer's identity,

[1]Source code available online: https://github.com/ahmed-pvamu/Location-based-Mobile-Advertising-System

financial status, and buying history with the LBS purchase parameters, including its exact time, place, purchasing behavior, and situational context, as it happened in real-time.

In transportation, Waze[7] can be considered as one of the largest community-orientated mobile travel applications with users volunteering information about their driving experience in real-time, by reporting on congestions, delays, and gasoline prices. These reports then become the basis for the information displayed on other drivers' maps (on their mobile devices), to help them make routing decisions. Similarly, TrafficPulse[8] combines location data from mobile devices with real-time traveler reports from frequent travelers and then offers this information to other drivers in an aggregate form.

In health, CrowdHelp[9] uses smartphones to collect direct feedback from mobile users about their medical condition, in combination with data coming from sensors in smartphones. This information is used to enable a swift response to emergencies. For example, when CrowdHelp is used for emergency reporting, mobile users submit information relevant to an event (such as the number of injured people and their state) to a central server. This information is collected and sent to the nearest health care facility capable of treating the injured.

Location-based services have also been used by Uga et al.[17] to build an earthquake warning system. The warning system uses data from both accelerometers and GPS present in many modern mobile devices to detect seismic vibrations. Devices send reports of likely seismic activity to a cloud server which then aggregates the reports received to send out warnings.

Our work is more closely related to research focused on supporting location-based advertising platforms. Existing efforts have taken different approaches to support such applications, focusing on concerns from programmability[18], to participatory crowd-sensing[19], to privacy[16]. We discuss these platforms below.

MobileAds[14] is a paid third-party ad serving platform for rich media and video advertising, which can be used by business owners to create sponsored campaigns. MobileAds allows businesses to create customized ads for a specific class of consumers. MobileAds then sends targeted ads to potential consumers based on their current locations, profiles (e.g., job occupation) and preferences (e.g., interests). However, as a paid service, it is difficult for small businesses to push ads to the most suitable customers with a limited budget.

Google AdWords[15] is an Ads platform that offers sponsored ads for businesses over various applications such as Google Search Engine and Maps. It promotes ads when customers search for what to do, where to go, and what to buy. Similar to MobileAds, AdWords is a paid service where businesses pay for every click to a business website or call the business phone number.

None of the existing privacy approaches for LBS systems can anonymize the mobile user trajectories while keeping a small energy footprint to the best of our knowledge. For instance, Hasan et al.[6] proposed a privacy architecture to preserve user trajectories in reward-based LBS applications that can anonymize the user trajectories with a fixed global location. However, the proposed approach suffers from a significant processing overhead when it is used in real-world applications. Another privacy-aware Location-Based Service (LBS) is proposed in[5] for enabling users to control the exposure of their location information securely. However, the proposed method sacrifices the service quality, such as result accuracy and computation and communication costs.

In summary, most of the existing frameworks for location-based advertising are paid services, which can not be affordable by small-to-medium businesses. Furthermore, they focus on investigating customer attitudes[3] and analyzing the business models[4]. However, none of the existing works studied designing an affordable LBS platform which can match owners and customers having budget-constrained businesses with multiple ad types to select. Also, none of them support concurrent execution of multiple location-based services from within one platform, which precludes taking advantage of opportunities to optimize for shared sensing requirements.

3 | SYSTEM DESIGN

As illustrated in Figure 1, the developed distributed system which is organized with parts executing on application and database servers hosted on the cloud, as well as on the users' mobile devices. The communication between the different sides is coordinated through a Representational State Transfer (REST) service layer. REST is a software architectural style that is used to provide interoperability between heterogeneous computer systems connected through the internet.

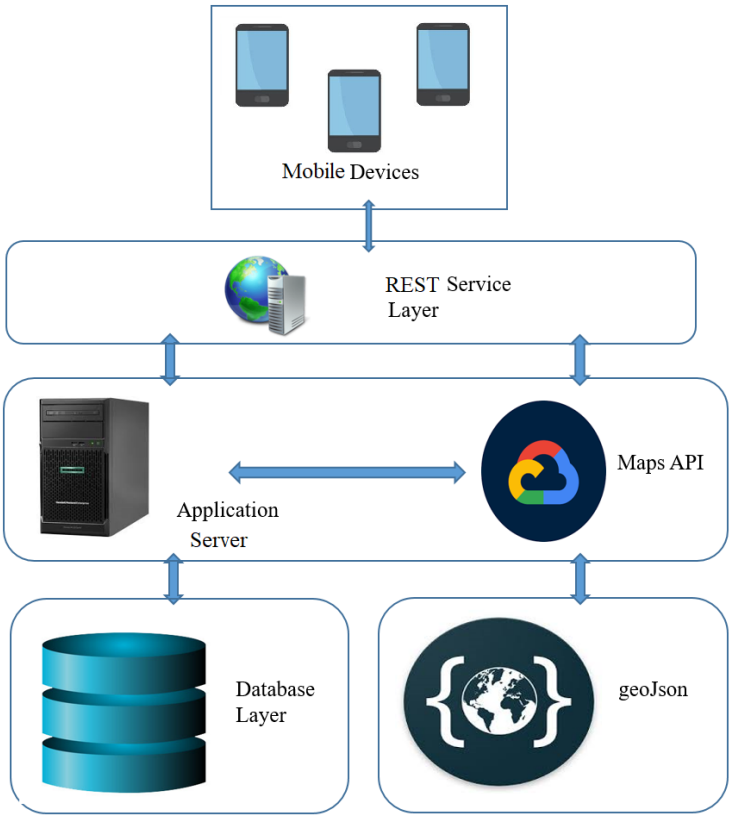

FIGURE 1 System Architecture

3.1 | Mobile User Side

At the user side, the mobile app is designed based on the multi-modal context-aware sensing concepts as discussed earlier in Section 1. The main idea behind multi-modal sensing is to program mode transition concerns of multi-modal mobile sensing applications separately from their functional concerns, which would otherwise be (and are) mixed.

In this paper, we are interested in modeling the context switching of mobile advertising systems, in which it is represented as a Finite State Machine (FSM) model. Using the proposed FSM model, we aim to model the dynamicity of the context-aware mobile advertising systems that adjust their own behavior according to changes in the context. Particularly, this model allows us to specify the modes in which a sensing-based application can be, the sensing needs of each mode, and the events which can trigger mode transition based. The model then monitors for mode transition events, and dynamically adjusts the sensing frequencies to match the current mode's requirements.

A natural way to represent multi-modal sensing is by using a finite state machine. Particularly, we specify a machine:

$$\langle M, \Sigma, \delta, M_0 \rangle \tag{1}$$

where M and Σ are non-empty finite sets of states and inputs respectively, with $M_0 \in M$ being the initial state.

Each state $M_i \in M$ represents a mode that a sensing device behaves at a certain time. The elements of Σ set are mainly: sensor readings, an observation of a change in context, or an observation of an interesting event such as approaching a business location. δ: $M \times \Sigma \rightarrow M$ is the state transition function. Σ contains triggers for mode change, defined in terms of recently sensed data. As shown in Figure 2, sensing of specific data can fire a trigger $t_i \in \Sigma$ which leads to state transition from M_i to M_j.

3.2 | Business Owner Side

At the business owner side, the web application utilizes GeoJSON for representing the geographical locations of businesses. GeoJSON is an open standard format designed for representing the geo-spatial information of places based on JavaScript Object

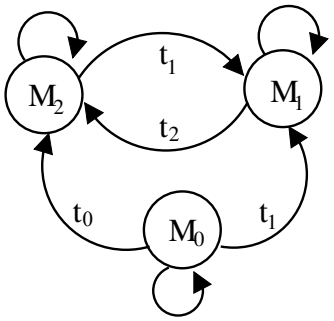

FIGURE 2 Multi-Modal Sensing

Notation (JSON). Each GeoJSON object represents a set of features of a place location, including the multi-polygon geometry type of the place which has its coordinates, size, and boundaries. Specifically, each place is represented by an array of polygon coordinates that states its boundaries on the map. Note that the starting point of the polygon coordinates is the same as the ending coordinate in order to close out the polygon loop.

Business owners have the ability to precisely select the polygon coordinates of their business locations using a user-friendly web interface. These coordinates are then stored into a database hosted on a cloud server. Once a new business location is created, the corresponding polygon coordinates are communicated to the mobile user side via the REST service layer in the form of a GeoJSON file.

When a new GeoJSON file is received at the mobile side, the mobile app runs a python script to parse that file for generating the place polygon coordinates, which is then fed to the Google Maps API for rendering the place on the interactive map. Figure 3 shows an example of a GeoJSON file communicated between the business and mobile sides. As shown in the figure, a GeoJSON object is defined to represent the spatially bounded entity of the place. The GeoJSON object represents the following elements: (i) a geometry type of `polygon`; (ii) a collection of geometries which contains the `coordinates` of the `polygon`; and (iii) a collection of features such as `properties`, `styleMapHash`, `description`, `fill-opacity`, etc.

```
geo_json = {"type": "Feature", "geometry": {"type":
"Polygon", "coordinates":[(30.095319, -95.993581),
(30.096340, -95.993399), (30.096015, -95.991854),
(30.095040, -95.992219), (30.095319, -95.993581)]},
        "properties": {"name": " Pizza Restaurant No 1",
           "styleUrl": "#poly-4F2682-3000-128",
           "styleHash": "-50cd947a",
           "styleMapHash": {
              "normal": "#poly-4F2682-3000-128-normal",
              "highlight": "#poly-4F2682-3000-128-highlight"
           },
           "description": "Pizza Restaurant location",
           "stroke": "#4f2682",
           "stroke-opacity": 1,
           "stroke-width": 3,
           "fill": "#4f2682",
           "fill-opacity": 0.5019607843137255
    }
}
```

FIGURE 3 An Example of a GeoJSON File

3.3 | Anonymization Module

The main objective of the proposed anonymization module is to camouflage the mobile user trajectories to ensure that the collected sensing data are communicated and stored safely. To achieve this objective, we developed a bounded perturbation method based on the GLS strategy. The rest of this section describes our anonymization modules formally.

Let u denotes a mobile user registered in our LBS system. Any movement of u results updating a tuple $< id, (x, y, t) >$, where id represents the identity of u, and $(x, y, t) >$ represents a visited point (x, y) by u at a specific time t, where x and y are the longitude and latitude, respectively.

The history movement of u can be used to drive the movement patterns of the system users. An identified trajectory L is defined by a sequence of successive Points of Interest (PoIs) visited by u over a period of time t, which is represented as follow:

$$L = \{id, (x_1, y_1, t_1), (x_2, y_2, t_2), \dots, (x_n, y_n, t_n)\} \tag{2}$$

where id is the identity of u, (x_i, y_i) represents the longitude and latitude of the PoI_i, and t_i is the corresponding time.

A GLS tuple consists of all n PoIs in a region, where every PoI has a location and description for its semantic meaning, such as walkway, highway, residential house, lake, mountain, and landmark. A GLS is represented as:

$$GLS = \{(coord_1, loc_1), (coord_2, loc_2), \dots, (coord_n, loc_n)\} \tag{3}$$

where $coord_i = (latitude_i, longitude_i)$ is a pair of coordinates of PoI_i, and loc_i represents its semantic description.

Algorithm 1 summarized the perturbation process for constructing the GLS tuples from a given map bounding box B. The algorithm describes the process of generating all possible visited locations within B's boundary.

Algorithm 1 GLS Generation Algorithm

```
1:  procedure GENERATING THE VISITED LOCATIONS
2:  Input: A Map Bounding Box (B)
3:  Output: The Global Location Set (GLS)
4:  /* Initialize B, GLS, visited locations V within B */
5:  B = {(x1, y1), (x2, y2)}; GLS = {};
6:  V = {highway, lake, resturant1, etc.};
7:  /* finding the valid locations in V */
8:  for each Point p ≡ (coord, loc) in B do
9:    /* if a visted location is found, add it to GLS */
10:   if (p(i) == V(j)) then
11:     GLS.add(p(i)) ;
12:   end if;
13: end for
14: return GLS;
15: end procedure
```

Our bounded perturbation method has to consider the environment noise r to ensure that the generated PoIs would not include invalid places in the interactive map. r is defined as:

$$r = (rd, ra) \tag{4}$$

where rd is a random distance in $l \leq rd \leq k$, l and k are the minimum and maximum distances (in miles), respectively, which depends on the area of the city, and ra is a random uniform direction calculated over a range of $[0, 2\pi]$.

Here is the definition of the generated PoIs $(\bar{x}_i, \bar{y}_i)$ after adding the random noise r:

$$\bar{x}_i = x_i + r_x \quad , \quad \bar{y}_i = y_i + r_y \tag{5}$$

where r_x is the random noise for x_n and r_y is the random noise for y_n.

To calculate random noise r_x and r_y for the longitude and latitude points, respectively, we used the Earth's radius R as follows:

$$r_x = \frac{rd}{R(\cos(\pi \times \frac{y_n}{ra}))} \times \frac{ra}{\pi} \tag{6}$$

$$r_y = \frac{rd}{R(\sin(\pi \times \frac{x_n}{ra}))} \times \frac{ra}{\pi} \tag{7}$$

Given an identified trajectory L, we apply our bounded perturbation method to generate the anonymized trajectory $\bar{L}$, which is represented as:

$$\bar{L} = \{id, (\bar{x_1}, \bar{y_1}, t_1), (\bar{x_2}, \bar{y_2}, t_2), \ldots, (\bar{x_n}, \bar{y_n}, t_n)\} \tag{8}$$

where id represents the identity of the mobile user u, $(\bar{x_1}, \bar{y_1})$ is the generated longitude and latitude PoI_i, and t is the corresponding time.

The generated anonymized trajectory $\bar{L}$ must be bounded by the GLS supported with the visit timing information of L. It also excludes the original visited places from the identified trajectories. Consequently, all anonymized trajectories $\bar{L}$s will protect the user's PoIs, unique routes, and frequent routes from public exposure.

We also calculate the relative distortion metric that measures the quality of the anonymized dataset compared with the original one. The relative distortion is calculated by finding the absolute difference of original and anonymization counts, as follows:

$$Distortion = \frac{|\beta - \Phi|}{\beta} \tag{9}$$

where β is the original dataset count and Φ is the anonymized dataset count. For example, if a PoI is visited 100 times in β and 60 times in Φ, then the relative distortion of the PoI is calculated as:

$$Distortion = \frac{|100 - 60|}{100} = 0.4 \tag{10}$$

Algorithm 2 summarized the process of anonymizing each point in the identified trajectory L, and generating the anonymized trajectory $\bar{L}$ within GLS. The algorithm takes the GLS and the identified trajectory (L) as input, and it produces the anonymized trajectory $\bar{L}$ in GLS as an output. L contains all user-visited locations.

Algorithm 2 Trajectory Anonymization Algorithm

1: **procedure** ANONYMIZING TRAJECTORY POINTS
2: **Input 1:** The Identified Trajectory (L)
3: **Input 2:** The Global Location Set (GLS)
4: **Output:** The anonymized trajectory $\bar{L}$ in GLS.
5: */* Initialize GLS_r, $\bar{L}$, and random noise r */*
6: $GLS_r = \{\}$; $\bar{L} = \{\}$; $r = (rd, ra)$;
7: */* Finding the actual visited locations in GLS */*
8: $GLS_r = \|GLS - L\|$;
9: */* Anonymize each point (r_x, r_y) in L */*
10: **for each** Point $p \equiv (coord, loc)$ **in** B **do**
11: $\bar{x} = \frac{rd}{R(\cos(\pi \times \frac{y_n}{ra}))} \times \frac{ra}{\pi}$;
12: $\bar{y} = \frac{rd}{R(\sin(\pi \times \frac{x_n}{ra}))} \times \frac{ra}{\pi}$;
13: $\bar{L}.add($**new** `tuple`$\{p.id, (\bar{x}, \bar{y}, t)\})$;
14: **end for**
15: **return** $\bar{L}$;
16: **end procedure**

First, we initialize the temporary GLS list (GLS_r), the anonymized location trajectory ($\bar{L}$), and the random noise (r). Second, we find the actually visited locations in GLS by subtracting the original visited locations in L from GLS to assure that the generated locations do not have any original visited locations by calculating their Euclidean distance. The result is stored in the temporary GLS list (GLS_r). In lines 10-14, we generate the anonymized trajectory $\bar{L}$ by anonymizing each point (r_x, r_y) in L

through adding the adding r. In line 13, the identity (id) and timing information (t) are added with the anonymized trajectory. Finally, the algorithm returns $\bar{L}$ in line 15.

4 | IMPLEMENTATION

This section presents the prototype implementation of the mobile advertising system which is divided into parts executing on the mobile devices and cloud servers.

4.1 | Business Owner Side

The business side is implemented as a web-based application that provides a customized dashboard for business owners to create a new ad, edit an existing ad, view applications in response to their ads, and approve/reject requests. The business side is developed using PHP 7.4, MySQL 8.0, HTML5, CSS3, JavaScript, and JSON.

All web pages are designed to be device-agnostic that can accommodate visitors using mobile devices, desktops or televisions to visit the web site. We used Bootstrap[20] to build a responsive, mobile-first and user-friendly web application for enhancing the business owner experience in using the system. The web application is loaded asynchronously with respect to the Google Map component which takes more time to load because the Map API key must be verified by Google before fully rendering the map on our web page.

Business owners must be registered in the system to get access to the ads dashboard. User credentials are verified and authenticated on each page using PHP Sessions. The PHP Session provides a unique number used to identify every user in an HTTP timeout session to ensure continued verification and authorization of the system.

Figure 4 shows a screenshot of the registration form at the business owner side. We need to gather a minimal amount of information from the business owner. Specifically, we asked the user's full name, email address, phone number, and the desired username and password. Indeed, we could ask for more information at this point, but a long-form is always a turn-off for many people. We also put some form validation code in place, so that we make sure that we have all the data required to create the user account. We check if the name and email, and password are filled in and that both the email address and phone number are in the proper format.

We used the `$_POST` method to send the data from the user browser to the application server on the cloud via an HTTP header. When the user fills out the form and clicks the submit button, the form data is sent for processing to a PHP file hosted at our application server, which in turn stores the user data into the database. In particular, an associative array of variables `$HTTP_POST_VARS`, which contains the form data, is passed to that PHP file via the HTTP POST method.

Similar to the registration form, we developed a login form that uses the `$_POST` method to verify and authenticate the users by matching their usernames and passwords with the ones from our database. If a correct combination was entered, we set a PHP session timed-variable for the user.

Figure 5 shows a screenshot of the page for creating a new ad at the business owner side. As shown in the figure, the form enables owners to draw a polygon on the map for specifying the boundary for their businesses. Besides the interactive map component, the user needs to enter: (i) the business name which will appear at the mobile app; (ii) the ad type which is an option from a dropdown list of various types of ads such as job, restaurant opening, garage sale, etc.; (iii) other relevant attributes including ad description, time frame, and business photos uploaded using a multi-photo uploader.

Figure 6 shows a screenshot of the list of ads created by one business owner. Using this form, business owners can view the status of their ads and edit/delete existing ads using a friendly interface. The form shows the business name, type, description, price, time frame, location, and geo-location.

4.2 | Mobile User Side

The user side is implemented as a self-contained mobile app developed using the Android Development Environment ADT bundle (64 bit). The app uses different technologies and tools including the Android SDK and XML to build the front-end activities, PHP 7.4 as a middleware between the app and the database server on the cloud, and MYSQL 8.0 for the database.

Figure 7 shows a snapshot of the landing page of the mobile app. The page has a search `textarea` control which enables the user to search and navigate for/to any destination supported by Google Maps. As shown in Figure 8, the search functionality

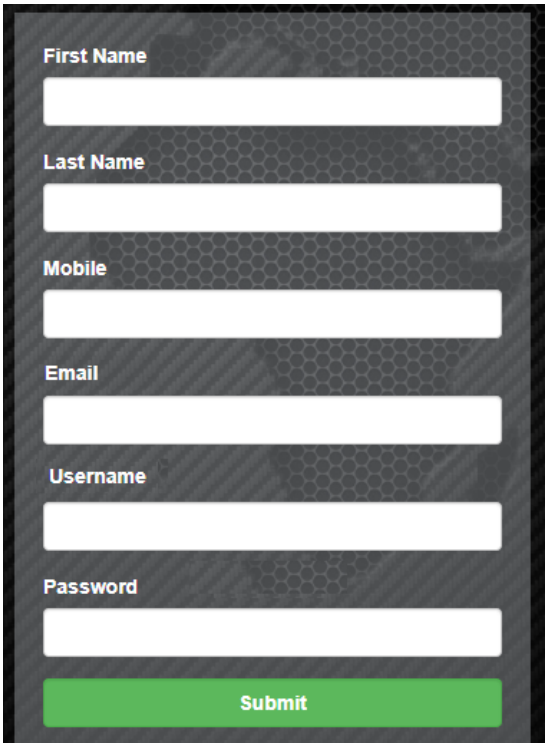

FIGURE 4 A Screenshot of the Registration Page at the Business Owner Side

supports the autocomplete option for increasing the user experience in using the mobile app. This functionality uses a predictive search algorithm based on popular Google searches to predict the user's search query as it is typed by providing a dropdown list of suggestions that change as the user adds more characters to the search input.

The user can navigate to the selected destination by clicking on the `START JOURNEY` button. The app then constructs the route between the user current location and the destination, while rendering the ads on the route between the two points. Figure 9 shows a screenshot of displaying the route map between the starting point and the example destination (`Addicks / Park Ten`).

Figure 10 shows a screenshot of displaying an ad in the driving mode at the mobile side. As shown in the figure, various types of ads are displayed on the map while driving between the starting location to the destination point. When the user clicks on an ad, another screen is displayed containing all ad details such as the ad description, timeframe, photos, etc.

We defined two types of class entities: Java Classes and Android Activities. A Java class represents a user defined set of properties or methods that are common to all objects of the class type. An Android activity represents a single screen of the app which contains its UI elements such as `ImageView`, `TextView` and `Button`.

The `LoginActivity` allows users to login using their Google accounts before directing them to the `InteractiveMap` screen where they can search for a destination. The `InteractiveMap` class is used to draw the route between the starting point and the selected destination by calling the appropriate Google Maps APIs. The `AdsDetailsActivity` represents the screen that displays the ad details including its description and photos. The `DirectionJSONParser` class is used to parse the geo-location of the destination and landmarks provided by Google Places API. The resulting data is loaded into a list `Routes`, which is an array of places, each of which contains a list of polygon coordinates that states the place boundaries on the map.

The `GPSTracker` class is used to obtain the user current GPS location defined by `LatLng` object. The `MapPhotos` class displays a list of ad photos on the map by fetching their URLs from the database. Both `MapsPoints` and `InteractiveMap` classes are responsible for displaying the user-defined ads on the interactive map. All ad details are sent to the map in a JSON format, which is then converted into Java objects using the Retrofit API[21]. Retrofit is a REST client for Java and Android which enables us to retrieve and upload JSON files in Android.

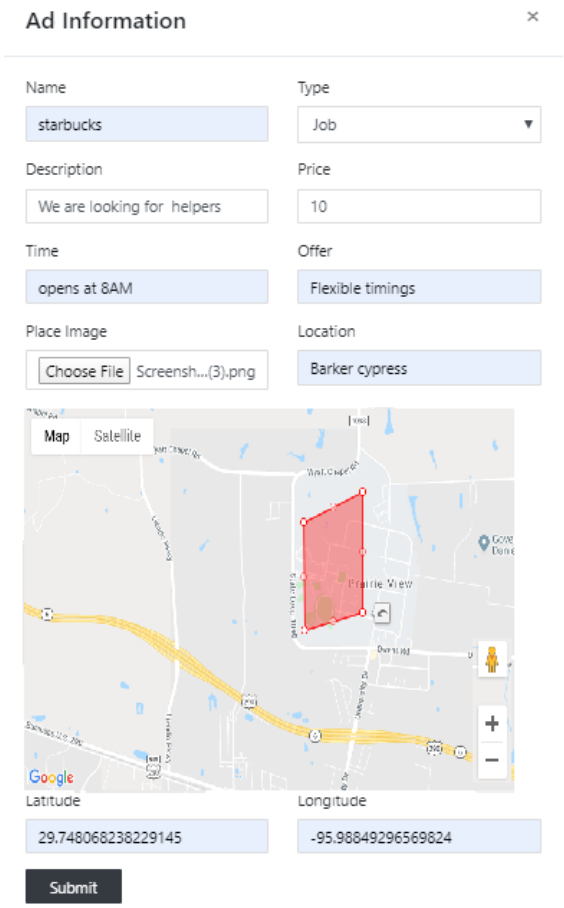

FIGURE 5 A Screenshot of the Page for Creating a New Ad at the Business Owner Side

4.3 | Multi-modal Sensing

The current implementation of the multi-modal sensing approach supports different types of triggers based on sensor feeds from the GPS, accelerometer and gyroscope. A set of high-level sensor triggers has been pre-implemented as a part of the mobile app in terms of these (low-level) sensor triggers –as executable specifications– which an application programmer can draw from and customize by providing parameters. These high-level triggers form the basis for the mode change of a multi-modal sensing application. For each high-level trigger feed, the list of required low-level sensor feeds is provided in the form of a list, where each entry identifies a sensor and specifies the rate at which it should be sampled.

Because the system will execute on battery-operated mobile devices, it is important that users have the ability to either develop or adopt simple resource consumption policies to avoid undesired battery drain. We have implemented a feature allowing the mobile user to specify resource limits after reaching which the device would stop consuming sensor resources.

5 | EXPERIMENTAL EVALUATION

We experimentally evaluated our prototype implementation regarding performance and scalability. We installed instrumentation in the mobile app running on a smartphone to measure the processor time taken to perform various tasks, including collecting sensor data, multi-modal sensing, GPS navigation, anonymizing user trajectories, etc. Instrumentation was also added to the mobile app to measure the energy costs of sensing GPS, accelerometer and gyroscope data. Each experiment presented in this section is carried out for ten trials, then we took the average of these trials' results.

Name	Type	Description	Price	Time	Location	Lat	Log	Edit
HydMoonCafe	Job	waiter	$11	opens at 9 AM	prairie view	28.10105	-97.32788	
starbucks	Resturant	restaurant	$11	opens at 10 AM	houston	28.64238	-98.95385	
Subway	Resturant	Authentic Indian restaurant, hyderabadi biryani	$30	opens at 11AM	prairie view	30.08740	97.327881	
starbucks	Job	We are looking for a waiter	$10	opens at 11AM	Katy, houston	30.08740	-98.953857	

FIGURE 6 A Screenshot of the List of Ads at the Business Owner Side

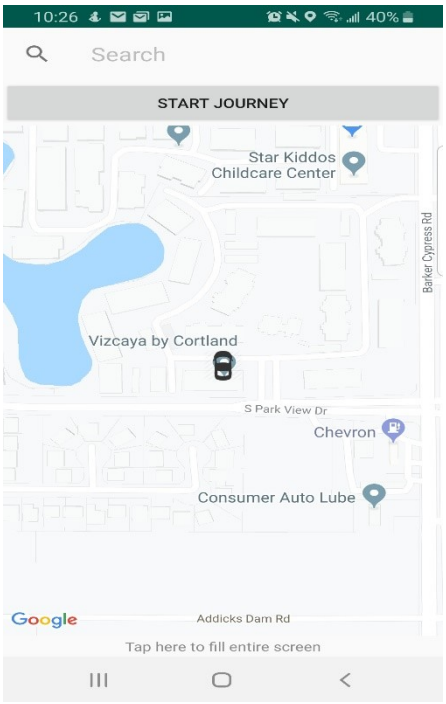

FIGURE 7 A Snapshot of the Landing Page of the Mobile App

Our experimental evaluation used a specific case study involving human activity recognition as a sensing application to simulate the different modes/activities for a user using our mobile app to view ads. The app ran on a Samsung S8 (2.3GHz Octa-core CPU, 4GB RAM, Adreno 540 GPU, LTE Category 16) running Android 10. We defined four activity modes, namely stilling (i.e., being still), walking, bicycling and driving. Each mode had its sensing requirements for one or more of the accelerometer, gyroscope and GPS sensors, which represented the functional requirements for the mobile application. The stilling mode required accelerometer data, walking mode required accelerometer and gyroscope data, and the bicycling and driving modes required accelerometer and GPS data.

In addition, the mode transition logic also relied on sensed data. Although it is not required in general, here we assumed that all sensors were to be sampled to detect mode transition triggers at a sampling rate of $1Hz$. In other words, for the sensors being

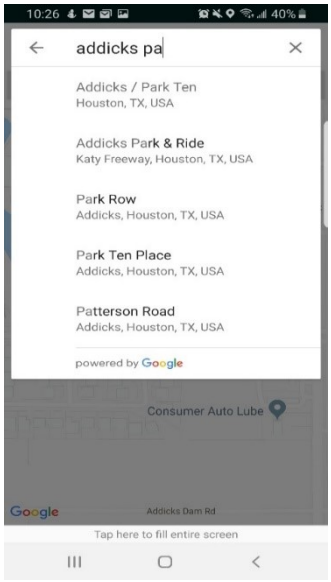

FIGURE 8 A Snapshot of a Mobile User Selecting a Destination Place at the Mobile Side

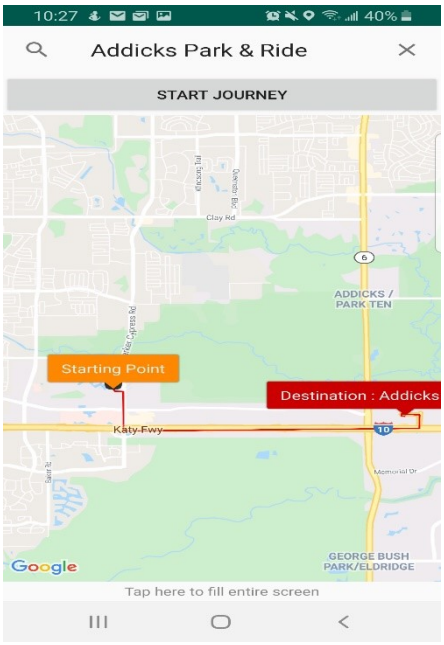

FIGURE 9 A Snapshot of displaying the Route Map at the Mobile Side

already sampled for the current mode's functionality, only one of a presumably larger number of samples was accessed every second for trigger detection purposes; for the remaining sensors, new sensing was required. The latter could also be pulled from sensing done for other applications as we have shown in[22].

5.1 | Performance

To evaluate the scalability of the mobile side, we measured the resources required to initiate and host a single ad. We separately measured the ongoing cost of monitoring for detecting mode transition triggers, and the sensing and processing delays in carrying out the triggered mode transitions.

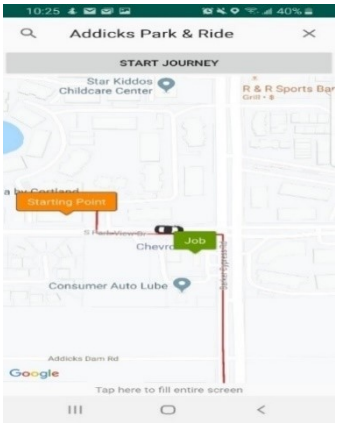

FIGURE 10 A Snapshot of displaying an Ad in the Driving Mode at the Mobile Side

TABLE 1 Ongoing Cost Per Feed Set of Monitoring for Mode Transition Triggers (in *ms*)

Mode	Accelerometer	Gyroscope	GPS	Total
Stilling	Old: 0.39	New: 4.50	New: 7.31	12.2
Walking	Old: 0.39	Old: 0.39	New: 7.31	8.09
Bicycling	Old: 0.39	New: 4.50	Old: 0.39	5.28
Driving	Old: 0.39	New: 4.50	Old: 0.39	5.28

TABLE 2 Mode Transitions Delay (in *ms*)

	Stilling	Walking	Bicycling	Driving
Stilling	X	12.32	19	19
Walking	12.32	X	31.32	31.32
Bicycling	19	31.32	X	1.61
Driving	19	31.32	1.61	X

The trigger detection mechanism checked for a trigger on arrival of every new sensor feed set, and examined a window of recently sensed data to detect a trigger. We found the trigger detection cost to depend primarily on the size of the window of recently sensed data considered. For our case study, we used a window of size 12, which seemed sufficient for detecting mode transition triggers.

Table 1 shows the processing time measured for the ongoing cost of monitoring for mode transition triggers for each of the modes. The time (in milliseconds) is for both acquiring and processing the set of feeds (one feed from each sensor) to check for a transition trigger. *Old* applies when data already being collected for the mode's function can be utilized; *New* applies when fresh sensing is required. Because the sensing for detecting triggers was at the rate of 1Hz in our case study, this cost is also in milliseconds per second. The delay in transitioning individual sensors from a current sampling rate to a different one were measured to be 6.21*ms* (*sd:* 1.21), 10.71*ms* (*sd:* 1.76) and 17.39*ms* (*sd:* 2.36) for the accelerometer, gyroscope and GPS sensors, respectively.

Table 2 shows the cost of making a transition from one of the four modes to another. This involved the sampling rate changes for the sensor involved plus a small amount of processing cost required to call for the changes. The sampling rate change cost was incurred only if a previously unsampled sensor was to be sampled in the new mode, or vice versa; the delay between a pair of modes is symmetric.

TABLE 3 The Energy Consumption of Sensors for Multi-Modal Sensing in mJ

	Accelerometer	**Gyroscope**	**GPS**
Cost Per Transition	2.42	3.18	4.05
On-Going Cost Per Feed Set	0.63	1.24	1.93

To put this on-going cost in perspective, on average, about 28 modes per second could be handled on a mobile device of our configuration (assuming no other computations are executing). If an average application requires as many as 10 data samples per second (from a variety of sensors), 2.8 of such applications could be supported; if an average of 1 data sample per second is required per application, a more likely scenario, 115 applications could be simultaneously running on a single mobile device.

5.2 | Energy Consumption

A set of experiments was carried out to measure the ongoing energy cost incurred by the sensors for additional sensing for detecting mode transition triggers, as well as the cost of change the sensors' sampling rates to carry out the mode change.

As shown in Table 3, the ongoing costs were measured to be $0.63mJ$, $1.24mJ$ and $1.93mJ$ for the accelerometer, gyroscope and GPS sensors respectively. These per feed set costs amounted to mJ per second for our case study, because the sensing for detecting triggers was at the rate of 1Hz. The cost of changing a sensor's sampling rate was measured to be $2.42mJ$, $3.18mJ$ and $4.05mJ$ for the accelerometer, gyroscope and GPS sensors, respectively.

In any reasonable scenario for mobile advertising, we would expect there to be only one mode transition every couple of seconds, which would not be a significant source of energy consumption.

5.3 | Overhead Analysis

To determine the non-sensing overhead of the multi-modal sensing approach, we measured the energy consumed by the mobile app required for mode transition, albeit without the actual sensing. The average energy consumed was measured to be 70.4 mJ for the accelerometer, and a similar 79.6 mJ for the gyroscope sensor. In percentage terms, this was roughly 4% of the total energy consumed in the accelerometer experiments, and 0.8% for the gyroscope sensor, the difference explained by the order-of-magnitude larger overall energy demand of the gyroscope sensor itself.

6 | CONCLUSIONS

This paper presented a privacy-preserving LBS-based system that helps small-to-medium businesses to advertise various types of ads affordably. It is expected that the developed system would create a better opportunity for small business owners to target potential customers with location-specific advertising on their smartphones. We developed two different types of applications as part of a distributed system executing on cloud servers and users' mobile devices. These applications allow users to initiate and manage various types of ads, making the system widely useable for different kinds of businesses. Furthermore, the paper presented an approach to programming context-aware sensing of multi-modal mobile applications separately from their functional concerns, leading to more modular code. The mode transition logic can be easily specified using an appropriate finite state machine.

To preserve the mobile users' privacy, we developed an anonymization module that proposes a novel bounded perturbation method to anonymize user's trajectories. The developed anonymization module can protect the mobile user's points of interest, unique routes, and frequent routes from public exposure. Thus, it could confront various types of inference and linking attacks.

We carried out several sets of experiments for evaluating the performance and scalability of our mobile side, paying particular attention to establishing the relationship between the number of ads hosted on the mobile app and the response time of the system. Specifically, we presented experimental results documenting the processing overhead, sensing delays and energy costs involved in using the mobile app to host a single ad, as well as achieving mode transitions. The experimental evaluation showed that the system is highly responsive and scalable despite the number of ads hosted in the system.

We expect that this research would increase the open-source knowledge base in the area of LBS-based mobile systems by publishing the source codes to the public domain. The source code available online: https://github.com/ahmed-pvamu/Location-based-Mobile-Advertising-System.

In on-going work, we are examining the composition of our system with ShareSens [23] to support the sharing of data between applications with multi-modal sensing requirements. The problem of cumulating demand on sensors is a significant barrier to simultaneously serving multiple sensing applications on battery-powered devices. Our work addressing this challenge has the potential of making it possible to simultaneously serve multiple sensing-heavy applications on a single mobile device.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available at: https://github.com/ahmed-pvamu/Location-based-Mobile-Advertising-System

CONFLICT OF INTEREST

Authors have no conflict of interest relevant to this article.

References

1. Ketelaar P, StefanBernritter , Riet J, Hühn AE. Disentangling location-based advertising: the effects of location congruency and medium type on consumers' ad attention and brand choice. *International Journal of Advertising* 2017; 36(2): 356–367.
2. Afanador IJJC, Rivero AJL, Gallego JR. Analysis of geolocation accuracy by GPS: dedicated support signal integration and collaborative network in location-based services. In: *Proceedings of the Iberian Conference on Information Systems and Technologies (CISTI)*; 2020: 1-8.
3. A Comparative Study on Attitudes Towards SMS Advertising and Mobile Application Advertising. *International Jounral on Mobile Communication* 2017; 15(5): 514–536.
4. Dhar S, Varshney U. Challenges and Business Models for Mobile Location-based Services and Advertising. *Communications of the ACM Magazine* 2011; 54(5): 121–128.
5. Ma C, Yan Z, Chen CW. SSPA-LBS: Scalable and Social-Friendly Privacy-Aware Location-Based Services. *IEEE Transactions on Multimedia* 2019; 21(8): 2146–2156.
6. Hasan ASMT, Qu Q, Li C, Chen L, Jiang Q. An Effective Privacy Architecture to Preserve User Trajectories in Reward-Based LBS Applications. *ISPRS International Journal of Geo-Information* 2018; 7(2): 1–16.
7. Waze: A GPS Navigation Software App owned by Google. accessed February 10, 2021.
8. Li RY, Liang S, Lee DW, Byon YJ. TrafficPulse: A Mobile GISystem for Transportation. In: *Proceedings of the ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*MobiGIS '12. ; 2012: 9–16.
9. Besaleva L, Weaver A. CrowdHelp: A crowdsourcing application for improving disaster management. In: *Proceedings of the IEEE Conference on Global Humanitarian Technology*GHTC '13. ; 2013: 185–190.
10. Zook M, Graham M, Shelton T, Gorman S. Volunteered Geographic Information and Crowdsourcing Disaster Relief: A Case Study of the Haitian Earthquake. *The Journal of World Medical and Health Policy* 2010; 2(2): 7–33.
11. Paulino D, Reis A, Barroso J, Paredes H. Mobile devices to monitor physical activity and health data. In: *Proceedings of the 12th Iberian Conference on Information Systems and Technologies (CISTI)*; 2017: 1–4.
12. PokemonGo: An Augmented Reality Mobile Game. accessed February 10, 2021.

13. Gong W, Zhang B, Li C. Location-Based Online Task Assignment and Path Planning for Mobile Crowdsensing. *IEEE Transactions on Vehicular Technology* 2019; 68(2): 1772-1783.

14. MobileAds: Third-Party Ad Serving Platform for Rich Media and Video Advertising. accessed February 10, 2021.

15. AdWords: Grow business with Google Ads. accessed February 10, 2021.

16. Hu W, Kaabouch N, Apostal SFJ, Yang H. Location-aware mining for privacy-preserving location-based advertising. In: *Proceedings of the IEEE International Conference on Electro Information Technology (EIT)*; 2017: 569–574.

17. Uga T, Nagaosa T, Kawashima D. An emergency earthquake warning system using mobile terminals with a built-in accelerometer. In: *Proceedings of the IEEE Conference on ITS Telecommunications*; 2012; Taipei, Taiwan: 837–842.

18. Ahmed AA, Jamali N. CSSWare: A Middleware for Scalable Mobile Crowd-Sourced Services. In: *Proceedings of the 7th EAI International Conference on Mobile Computing, Applications and Services (MobiCASE '15)*; 2015; Berlin, Germany: 181–199.

19. Min M, Sasank R, Katie S, et al. PEIR: the personal environmental impact report, as a platform for participatory sensing systems research. In: *Proceedings of the ACM Conference on Mobile Systems, Applications, and Services MobiSys*MobiSys '09. ; 2009; Poland.

20. Bootstrap: An open-source framework for designing front-end web application. accessed February 10, 2021.

21. Retrofit: A type-safe HTTP client for Android and Java. accessed February 10, 2021.

22. Moamen AA, Jamali N. Opportunistic Sharing of Continuous Mobile Sensing Data for Energy and Power Conservation. *IEEE Transactions on Services Computing* 2017(doi: 10.1109/TSC.2017.2705685): 1–14.

23. Moamen AA, Jamali N. ShareSens: An Approach to Optimizing Energy Consumption of Continuous Mobile Sensing Workloads. In: *Proceedings of the 2015 IEEE International Conference on Mobile Services (MS '15)*; 2015; New York, USA: 89–96.