Cellular Networks and Mobile Computing
COMS E6998
Final Project Report

SHOPPING BUDDY

Team Members:
Neha Aggarwal na2542
Nikhil Raju nr2483
Rafica Abdul Rahim ra2688
Swikriti Jain sj2615

Table of Contents

Motivation:

People today are very busy and tend to forget what they need to buy. There are existing apps which allow the user to make a reminder list, but when a user is in the market, he/she needs to look up in the list. This app will give the users an alert of the stores where they can buy the stuff mentioned in the list, thereby reducing the effort to remember and manually search for stores for a given item. This way whenever a user goes to market, he/she will automatically get updates about the stores to shop various items.

Current Solutions:

There are apps which help you organize your shopping list but generally lack a "reminder" capability or if they have a reminder capability, the user has to set the reminder manually for a particular time. Whereas our app will automatically use the location data and remind the user in form of an alert about the items to buy from various stores in a radius around the user. Some of the existing apps such as "Shopping List" on the Google Play Store are limited to groceries. Our app aims to provide information for broader categories such as clothing, electronics,furniture

Problem Statement and Proposed Solution (Features of Application) :

- Allow users to make a checklist of items they wish to buy
- Organize into categories based on type of item such as groceries, clothing, apparel,electronics
- The user should be able to choose a priority High,Medium or Low for each item
- Ask for the permission of the user before switching on GPS
- Use the GPS location and find stores in a particular radius for the items on the shopping list
- Should provide an option to the user to disable notifications.

Alert the user when he is inactive for an extended period of time(say 30 minutes) and ask if he wishes to switch off GPS

Design and Architecture :

Concepts Applied from the Course:

Power consumption:
In any location based mobile application energy efficiency is crucial. To minimize the power consumed we have incorporated the following key features in the design:
We have made use of the accelerometer. When the user requests the app to point him or her to the desired location, the accelerometer registers a service and uses the device's sensors to point to the correct direction.
This service is unregistered when not in use to minimize the power consumption related to the device's sensors.
The frequency for location queries is decided by the priority of the list.
(what if there is even 1 high priority list for too many days and user forgets about it?)
The application requires GPS but always asks for the users permission before turning GPS on to prevent battery drain.
Notifies user if a WiFi network is available to reduce the data and power consumption.

Scalability:
The app should ensure that the performance does not degrade in any way as the number of users using the app simultaneously shoots up. There could be spikes and sudden drops in the query traffic and database requests. To ensure appropriate scalability, we have made use of SQLite.

Security:
It is important that a user be identified before access to the app is given. Also the app stores personalized data about lists added by the user. So a login access is required. For authentication, we have used Facebook OAuth. This eliminates the need for making the user manually enter information through forms. Also using SQLite to store the user's data addresses privacy and security concerns as both Facebook OAuth and SQLite are well established and address privacy and security concerns that the users may have.

Notifications:
The app makes use of System Notifications to notify user when he or she is near a store that sells items on the lists added by the user.

Implementation and Testing

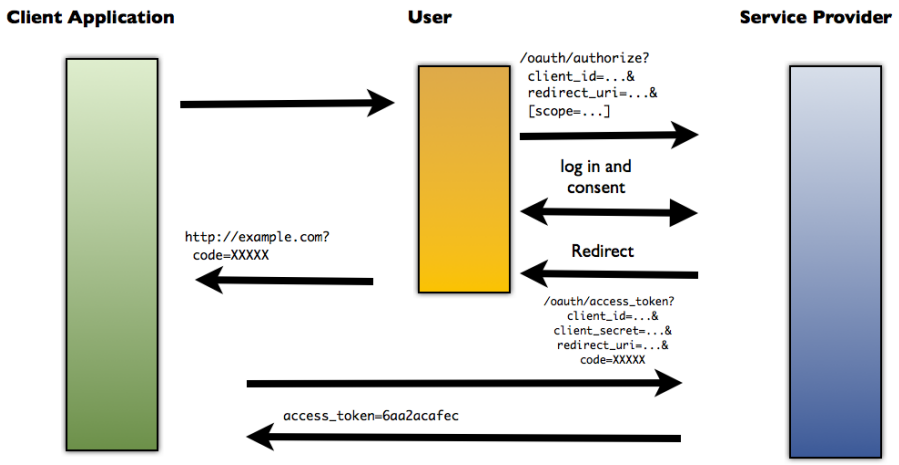
Google Maps API

We have used the Android Google Maps API to implement the location finding functionalities. The Maps API obtains the location i.e latitude and longitude of all the stores in a particular radius for the items on the shopping list and marks it on the map. The user's own location is displayed on the map. An API key is obtained to access the Google Maps. Using an API key enables you to monitor your application's Maps API usage, and ensures that Google can contact you about your application if necessary.

Facebook API

The Android Application needs a mode of authentication of the identity of the user. Here, API of Facebook is used for that purpose.
This is done as follows:

1. Required permissions are added in AndroidManifest.xml file.
2. Layout is created in res> main_oauth.xml
3. Initialize the source with necessary constants and create an Oauth listener.



SQLite

SQLite is an Open Source database. SQLite supports standard relational database features like SQL syntax, transactions and prepared statements. It is used as a way of storing user data. We are using SQLite to store the data about the lists of the user along with the list items, priority and the type.

Google Place search API

The Google Places API allows you to query for place information on a variety of categories, such as: establishments, prominent points of interest, geographic locations, and more. We are using this API to search for locations on the basis of the category of item such as Electronics, Clothing etc. This allows to look for the items in the list around the fixed radius of the user and display them on the map.

Android Accelerometer

The accelerometer is a hardware sensor used to detect a shake motion. When a user finds a store which he wants to visit and taps on the "point me to the location", the accelerometer is used to give direction to the user to reach that location.

Lessons Learned

What worked

We were able to implement the following features:
priority based checklists, addition of items based on category, a point me there feature using the accelerometer.
Feature to add a custom item not found in the items provided by the app.
Authentication using Facebook OAuth. SQLite database for persistence and to store data about user lists

What didn't work

The proposal included a plan to include some additional features for improved UI experience such as :
include favorite items to allow the user to easily add items that were added in recent lists.

Summary and Further Work

The ShoppingBuddy app was implemented successfully with most of the features specified in the project proposal.

To improve the application we plan to implement the following enhancements:

- Additional feature to store favorite items/ recently added items so that user can quickly add frequently bought items.
- We plan to optimize the power usage further by making use of the accelerometer to calculate distances within a short radius using the device's sensors rather than relying on GPS.

Contributors:

Neha Aggarwal:
Acted as an anchor to co-ordinate work with all other team members. Worked on the design of screens and focussed on the backend API's used to find data about stores and locations. Worked on the database design and integration with front end. Also contributed to implementation of the expandable lists adapter.

Nikhil Raju:
Focussed on the design of screens, navigation drawer and the UI. Implemented the AddItemToListActivity,DetailViewActivity, Expandable List Adapter and integration with backend database.

Rafica Abdul Rahim:
Implemented the login by integrating Facebook OAuth. Also implemented the "Point Me There" feature.

Swikriti Jain:
Focussed on the notifications provided by the app. Also contributed to the design of screens and navigation drawer for the home screen

References and Literature Reviewed

1. http://www.vogella.com/tutorials/AndroidSQLite/article.html

2. https://developers.google.com/maps/

3. http://www.research.att.com/articles/featured_stories/2011_03/201102_Energy_efficient?fbid=bj7IxEKH3ZE

4. http://www.techrepublic.com/blog/software-engineer/a-quick-tutorial-on- coding-androids-accelerometer/472/

5. http://developer.android.com/design/patterns/notifications.html