LocationReminders

Sarah Smith
Chenyun Zhang
Haoran Liu

Problem Overview and Introduction

(Comments from Prof.Jeff "Problem Overview and Intro (20) – 90%
Good problem overview. You may want to add more content to convince the reader that this is actually a problem. Some people may think that when you put together a list of things to buy, a person already has in mind where they need to go. By reading this introduction, I am not fully convinced that I would need this app. You should try and really emphasize that this is a significant problem. You may also want to bring up some adhoc events, where users may be close to the vicinity of a store that has something they need, but missed that opportunity because they did not know about it. ")

The following is to be revised….

Nowadays people's rhythm of life is becoming faster and faster. It's always difficult to remember what you need from different stores when you go out for shopping. What's more, sometimes you are not sure whether the desired store is nearby when you are in a place. And if you keep track of many to-do lists spread across different apps, the situation becomes worse.

We are now designing an app to solve the problems we discussed above perfectly. The app should includes the following main features:

1) Create different to-do lists per store, or type of store. Each store that you may go to corresponds to a to-do list, which lists the goods you wanna purchase.

2) To-do lists are localized in the same app. All the to-do lists will be stored in the same app, and you can manage them easily.

3) The app alerts you when you are close to a store that you need items from. The app has Location Based Services which are able to alert you when you are near a desired store. The standard of 'near' can be set by users to be anywhere from 1 block to .25 miles.

4) Work across different chain stores to find the closest one. When you select a store, which is a chain store, the app can show you the closest one. Similarly, if you request a type of store, such as a drugstore, it will show you information from stores such as CVS, Duane Reade, and Walgreens. This way, if you are not specifically needing one drug store over another, you can find your items faster.

Current Solutions and Related Work

(Comments from Prof.Jeff "Current Solutions and Related Work (10) – 80%
I would recommend researching different potential categories of apps that are similar to yours. Note and checklist applications are one category that you already mention (things like Google Keep and other note apps would fit into this category as well). There is also a lot of related work out there that has to do with location-based services and advertising. iBeacon is one possible mechanism for this. You should look around for other services that are similar or are used in different scenarios but utilize similar technology. Only 3 examples isn't enough. ")

The following is to be revised...
Currently, people are often using following apps with different functional defects that Location Reminders seeks to fix.

1) Evernote. It is easy to create checklists and synchronize across phone and website. You can choose from making checklists, notes, and to-do-lists. You can also share your notes with other people through facebook and twitter. For the pro version of Evernote, you can have the option to save notes offline, so they are always available. However, Evernote does not support the location-based alert service.

2) Memorix Notes+checklists. It allows you to coordinate checklists by different colors and share notes with other people. You can share across email and SMS, but not with facebook or twitter, which were options in Evernote. One advantage for Memorix is you can securely store notes in a

password-protected vault. Unfortunately, like Evernote, it is not able to provide you with an alert service.

3) iBeacon. It can be used to alert users when they are nearby stores or products within stores. Currently, it only exists on iOS. It works by pushing notifications to the iPhone when the phone is near the requested object. But it is not specifically designed for remind people when they are near the containing store, gps-wise. It only notifies once inside that store.

Solution

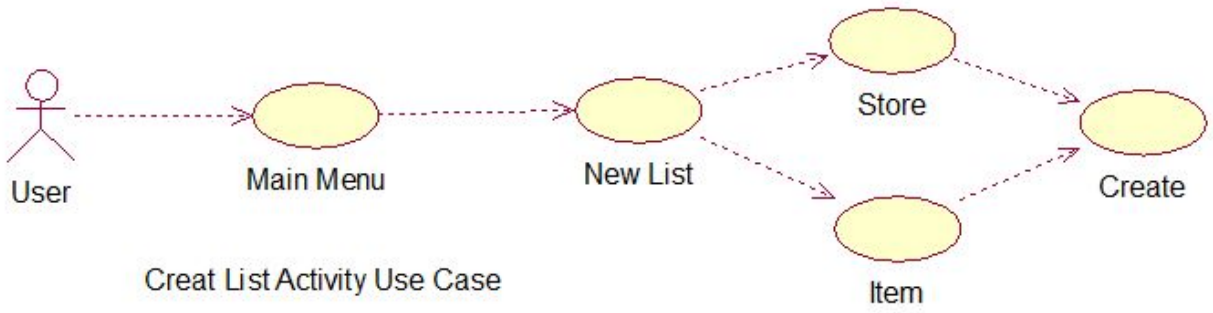
(Comments from Prof.Jeff "Solution (30) – 70%
There are many details that are missing from the solution description. Specifically, what algorithms are you planning to develop to "remind" the user that they are near a location where they can purchase something on their list. Also, there are no descriptions of how you are planning to make this app unique compared to the checklist and note applications you mention above. How are you planning to share this list with other users? As of right now, this design document makes your app look similar the other apps without any of the unique features that you mention in the beginning.
Good details on the models and controllers.
An interesting extension of this work would be to survey the nearby area based on what the user needs and let the user know that the product they need is cheaper at one store compared to another.")

The following is to be revised...

In order to create a working and usable Location Reminders app, we need to focus on how the users will interact with the app. It will start off with prompting users to either login or register to make an account. There will be a backend server for this regard. Once users login, they are able to create, view, and edit their to-do lists. They can also share them with friends via email, sms, facebook, or google+. Google maps is also incorporated into this app, so that users can view where the stores are on the map. They can also click on a store on the map and then proceed to create a list for that store. All lists can be stored both locally on the device, and on the backend server, so that users can view their lists across different platforms.

Use cases

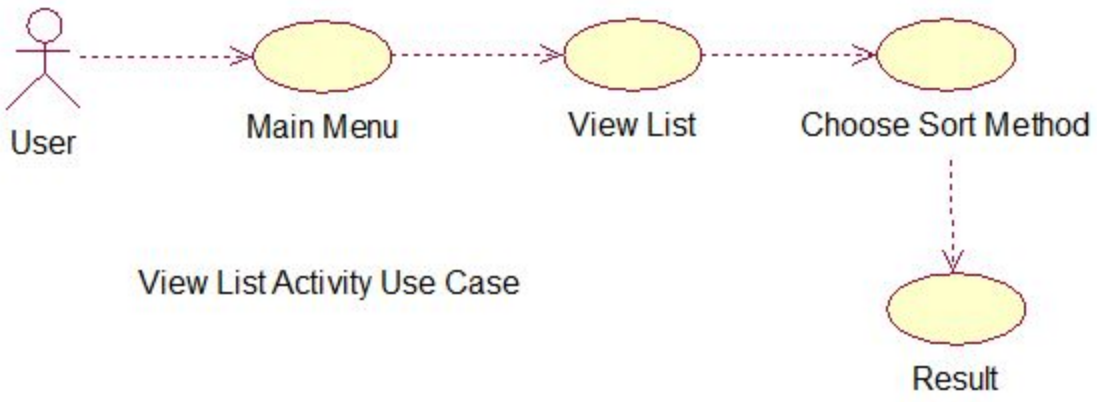
1. Create List



UC1_Creating new checklist
Description: Process for creating new checklists
Primary Actor: User
Pre-condition: User opened the application.
Basic flow: 1. User inputs a store for shopping 2. User inputs what to buy. 3. Press OK button. 4. Back to Home page to see the new to-do list. **Alternative Flows:** a. User cancels creating to-do list: 1a. Welcome page. b. User doesn't know exactly where to buy: 1b. User inputs what to buy. 2b. Search things through Internet and returns results. 3b. User chooses some results and adds to to-do list.

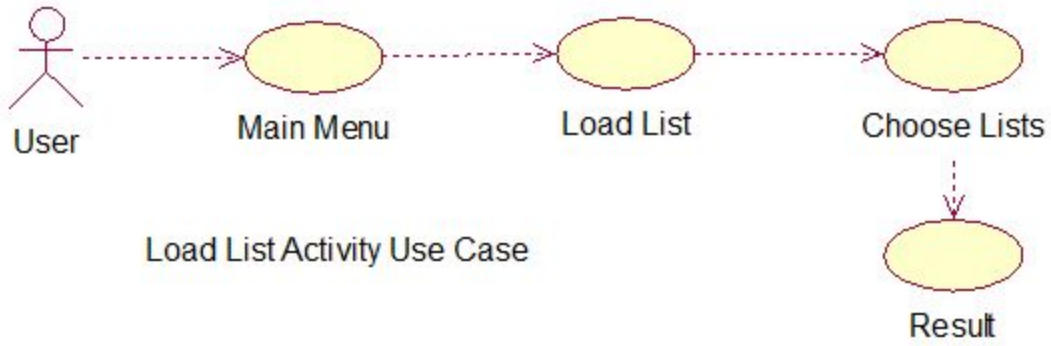
4b. Press OK button. 5b. Back to Home page to see the new to-do list.

2. View List



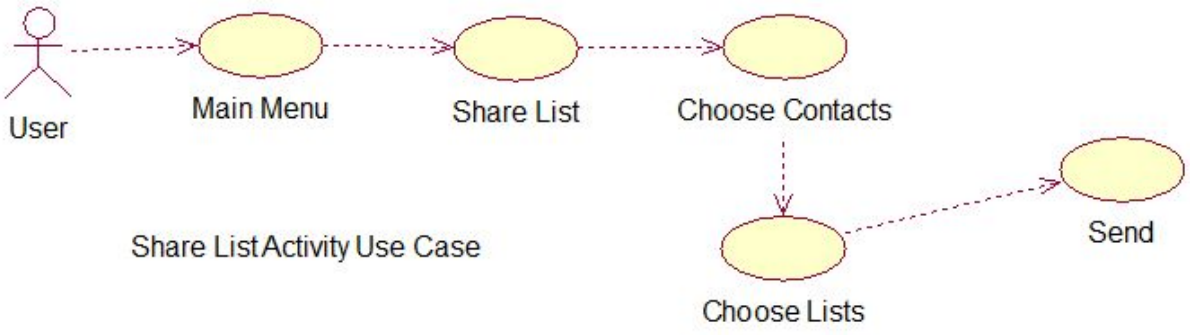
UC2_View list
Description: Process for viewing checklists
Primary Actor: User
Pre-condition: User opened the application.
Basic flow: 1. User click view list button. 2. View lists. **Alternative Flows:** a. User cancels view list: 1a. Home page. b. User empty checklists: 1b. User click empty button. 2b. Checklists cleared.

3. Load List



UC3_Load list
Description: Process for loading checklists
Primary Actor: User
Pre-condition: User opened the application.
Basic flow: 1. User clicks load list button. 2. User selects list by spinner. 3. Press OK button. 4. View items. **Alternative Flows:** a. User cancels load list: 1a. Home page. b. User chooses checkbox: 1b. User click checkbox to select items.

4. Share List



UC4_Share list
Description: Process for sharing checklists
Primary Actor: User
Pre-condition: User opened the application.
Basic flow: 1. User opens contact list of phone 2. User chooses friends to share list. 3. Press OK button. 4. Back to Home page to see the to-do list. **Alternative Flows:** a. User cancels sharing to-do list: 1a. Home page.

5. UC5_Notification

UI Drawings

Sarah, could you paste the last version APP screen shot here? We can't run the APP. Thank you.

MVC Framework

Model:

Item, Place.

Class Name: Item.
The Item Class will be used to store user's dream items' information.
Integer id; String item; String store; String geoLocation;

Calendar date=Calendar.getInstance();

Class Name: Place.
The Location Class will be used to store user's and item's GPS location.
String name; String date; String item; int id;

Views:

Welcome, Home, Create lists, Load lists, View Lists, Share Lists.

Class Name: Welcome
The WelcomeView Class will be used to display a welcome page to users.
Main_menu Button This Main_menu Button will lead users to go main menu.

Class Name: Home
The HomeView Class will be used to display a main menu page to users.
Create List Button This Create List Button will lead users to go create page.. Load List Button This Load List Button will lead users to go load lists page.. View List Button This View List Button will lead users to go view lists page.. Share List Button This Share List Button will lead users to go share lists page..

Class Name: Create Lists
The Create Lists View Class will be used to display a create lists page to users.
Add_Item Button This Add_Item Button will lead users to add more items. Create Button This Create Button will save new lists and lead users to go Home page.

Class Name: Load Lists
The Load Lists View Class will be used to display a welcome page to users.
Spinner This Spinner will lead users to select checklists. checkbox This checkbox will lead users to select items. OK_Button This OK_Button will lead users to go Home page.

Class Name: Share Lists
The Share Lists View Class will be used to display a share lists page to users.
Add_Contacts Button This Add_Contacts Button will lead users to go contacts and selects friends. Next Button This Next Button will lead users to select checklists page. Back Button This Back Button will lead users to Home page. Share Button This Share Button will share checklists with the selected friends.

Class Name: View Lists
The View List Class will be used to display a list view page to users.
Back Button This Back Button will lead users to go Home page. Empty Button This Empty Button will clear the data in SQLite. Spinner This Spinner will lead users to select sort methods. Click the list This Click will lead users to go lists detail view. Share Google+ Button This Share Google+ Button will share your lists in Google+.

Controllers:

WelcomeController, HomeController, CreateController, LoadController, ViewController, ShareController, AlertsController.

Class Name	**Welcome**
Need	The WelcomeController Class will be used to manage welcome page to users.

Class Name	**Home**
Need	The HomeController Class will be used to manage Home page to users.

Class Name	**Create**
Need	The CreateController Class will be used to manage Create page to items.

Class Name	**Load**

Need	The LoadController Class will be used to manage load page to items.

Class Name	**View**
Need	The ViewController Class will be used to manage viewlists page to items.

Class Name	**Share**
Need	The ShareController Class will be used to manage share page to items.

Class Name	**Alerts**
Need	The AlertsController Class will be used to manage alerts notification to users.

Other Android Components(if there has)
Login with Google account and share lists in Google+ circle.

Project Member Breakdowns

What Sarah E. Smith did:
1. GPS notification function: collecting GPS location and alerting users if stores in checklists are nearby.
2. Sharing function: logging into Google circle, sharing checklists with Google+.

What Haoran Liu did:
1. Create checklists function.

2. Load checklists with spinner and checkbox.
3. GUI designing.

What Chenyun Zhang did:
1. Sorting function: list view the checklists, view checklists' details, sort checklists by Date, Store, Item.
2. Sharing function: Sharing checklists with friends via SMS.

Goals for each Milestone

3/16/2014 – Initial Design Document

3/23/2014 -- Checklists almost fully implemented

3/30/2014 -- Checklist sharing, sort lists, gather gps data

4/7/2014 – Initial Prototype Presentation

4/14/2014 -- gps pop-up alerts, all checklists and sorting done

4/28/2014 – Presentation at AT&T

5/4/2014 -- Implement suggested changes from AT&T Presentation

5/12/2014 – Final demo

Implemented Feedback from AT&T

1. Log in with Google account.
2. Share lists with friends via Google+.
3. Finish location services and location notification.
4. More advanced GUI.

Future work

1. Server function: build a cloud server to save data.
2. Smart Search: type in items, app will recommend the store.
3. More advanced GUI.
4. Incorporate some more of the suggestions into the project