Proposal for a Thesis

In the field of Software Engineering

In Partial Fulfillment of the Requirements

For a Master of Liberal Arts Degree

Harvard University

Extension School

April 27th, 2017

David N. Rasoly

10th Reuven Rubin

Tel Aviv, Israel, 6941261

DavidRasuli@g.Harvard.edu

1.Tentative title

Shared Shopping List – A distributed shared shopping list application

2.Abstract

The goal of this project is to allow multiple participant management of a shopping list via an application called "Shared Shopping List" – a shopping list which will allow managing actual physical products via, and invite subscribers who already has the Shared Shopping List to act as a subscribers and participants to list from the phones contacts repository, using phone number as a unique identification.

3.Thesis Project Description

3.1. Background

Shopping list notes were and still are very popular tool for managing shopping and serve as a reminder while shopping for any product, from daily groceries to electronics and hardware materials. As mobile technologies has advanced many software applications such as "Don't Forget The Milk" , "AnyList" and "Keep" began to show and uses easy user experience along with the option to track the progress of the shopping.

The problem with these applications is that most of them (Google Keep is an exception), are for self-use only. Imagine a scenario when while in shopping, your partner found out that few things are missing from the original list, or that you, as a shopper , found out that some of the products are out of stock. Obviously a phone call could solve this, but if one's attention is already focused on the mobile's shopping list, it would be great if one could add and remove products and notify it to all the subscribers of the shopping list, especially if the buyer will conduct the shopping for an arranged group of shoppers.

Shared Shopping List should allow both the shopper and the subscribers to add, remove, modify and notify each other about suggested changes at the list. The shopper could let the other subscribers know what items are already at the cart and if a product is missing and could suggest replacements, all of that with the additional base functionality of both tracking and existing list, creating new lists, loading, saving and editing existing shopping list.

The goal of this project will be to adapt a cloud solution which will both serve as a products, users and shopping lists data management server, and both as an API that serves the end client mobile application.

In order to implement the idea successfully, the functionality of the project will contain three keys of success: cloud services, mobile integration, and simple UX/UI. Please see additional details on each key of success below.

Cloud Services:

Due to the distributed nature of the application, cloud services will carry two roles:

1. Products, Users and List management entities via a remote database.

2. API support which handles the functionality of the project. Since the client expect both notifications and ability to execute remote commands from the Server, the cloud services will provide these abilities by Web Methods and Push Notifications.

Mobile Integration

A client mobile application which will be connected to the distributed cloud services, and work against one of the major mobile operation systems, either IOS or Android.

Simple UX/UI

In order to keep the application usage as friendly and simple as possible, the application will contain as few screens as possible.

The aim is to have five UX Screens:

1. Entry screen – welcome, menu, about.

2. Configuration screen – allow/disallow notifications, invitation to other active lists.

3. Lists management screen – displays, loads, removes and saves shopping lists.

4. Active product list management – active product list management, shows each product's state and allows edit of products.

5. Participant management – removes and invites participants.

Please refer to section 5.2 to read further about the implementation and functionality of these screens.

3.2 Prior work

"Remember the Milk" – a mobile application that handles list item editing and reminder via scheduling systems. Besides lists management and scheduling, it provides item priority and tagging support, along with synchronization to Microsoft outlook. This application provides all the necessary lists and items management features that one wants, but is limited to self-use only.

"Google Keep" – a shared list management application. A user can create a list and edit list of items such as text messages, images, and audio. One or more participant can share the same list; all of the participants can edit the list simultaneously. "Shared

Shopping List" aims to distinguished its functionality and use to shopping items only – and will do so by allowing to show count of requested items, along with its availability and alternative item suggestion in case absent of a desired items.

4. Software:

Programming languages:

The Android core programming language is Java, so it will be wise to choose this programming language for Android client development. Amazon Lambda also supports Java as a programming language. However, due to extensive examples and guides for Amazon Lambda in Node.js, both or either could serve as the server's programming language.

4.1. Server:

Amazon Web Services (AWS) provides comprehensive cloud solutions, from databases, to remote machines, scaling systems, security, notifications, queueing, analytics and more. It has an administrative dashboard which ease the management of all of its desired services, making it a comfortable and easy to manage from one back end system.

Amazon Lambda – AWS Lambda will allow the server code to run without the need of managing the cloud resources, this allows automatic scalability and performance tuning.

Amazon SNS – Push notification support - some of the clients' functionality relays on notifications from the server, amazon SNS will provide this utility.

Amazon RDS – Data storage and management, will store and manage the Active Shopping List, Users, Accounts, Logs and Products.

Amazon Lex – Using speech recognition and natural language understanding, Lex allows creating smart chat boxes, this will be used to automate the help and Q&A section of the application.

4.2 Client:

Due to the reduced costs and ease of development comparing to IOS, and to prevent multiple languages and technologies in one solution, this project's client will be implemented using Android and will be developed with Android Java.

Android Visual Studio – a powerful IDE for development and deployment of Android mobile applications.

Android Device/Emulator – for client deployment and testing.

5.Functionality details:

The solution will follow the principles of Client-Server Architecture; Client will be implemented in the form of a friendly, easy to use mobile application, while Server side functionality and data storage will be fully implemented and deployed over the cloud.

The functionality assumes three domain entities:

Shopper – the actual person who does the shopping, he or she will act as the Admin of the Active Shopping List.

Subscriber – anyone other subscriber on the Active Shopping List – has a more limited modification rights on the list than the Shopper.

App – Suggestions, sharing, app invite and any other functionality which has indirect or no effect on an Active Shopping List.

5.1.Cloud services

Web methods

List domain:

Create new list. - Shopper entity.

Edit existing list. - Shopper entity.

Share list. - Shopper & Participant entities.

Finish shopping. Shopper entity.

Participants' domain:

Invite to list.

Remove from list.

Assign Shopper admin. - Shopper entity.

Item domain.

Create new item. - note that there is no edit on a new item; an edited item becomes a new item.

Add item to list. - Shopper & Participant entities.

Tag item as "Unavailable". - Shopper entity.

Mark item as "Added to cart". - Shopper entity.

App domain.

Invite to app.

Notifications:

Active List domain :

Missing item.

New item suggestion.

Item in cart.

New Shopper admin.

Participant added.

Participant removed.

List created.

List closed.

Artificial Intelligence:

Help section – opens a chat box intent(perform an action in response to natural user input), with pre-defined utterances(spoken or types sentence that invoke the intent), and slots (input data required to fulfill the intent).

5.2. Interactive

Following the Android development paradigms suggested by the official Android development documentations - the UI screens will be implemented as Android Activity, where each screen implements an activity of its own. The following operations of each activity will be carried via Android Intents, which serves as an abstraction around any UI operation to be performed.

Entry screen.

Log in – either via Google accounts or as a guest.

Create new shopping list.

Active product list management

Active shopping list screen. - A table with the following headers: Product Name, Company (optional), Quantity + Generic Measurement Unit (#, kg, lb., oz., etc) , Picture(optional).

Check and notify once a product has been found & added to the cart. - Shopper entity.

Check and notify a missing product. - Shopper entity.

Add and notify about a new product. - Shopper and Participants.

Suggestion from existing product repository (App).

Finish Shopping.

Participant management

Invite participants.

Remove participants.

Lists management screen :

Save/Load/Edit/Delete a shopping list.

Configuration screen.

Allow invites.

Allow notification from other users in shopping list.

Allow product suggestions.

5.Implementation Limitations.

Due to extensive learning curve, design and implementation, the client will only support Android users.
All of the cloud services will be limited to "Free Tier" deal, which limits the scale, database request throughput capacity, and usage of Lambda Services.

Please see section 8 for more comprehensive details about the "Free Tier" deal.

<u>6.Project Milestones</u>

<u>Discovery & Design: 6-8 weeks.</u>

Discovery will include the following proof of concepts (POC):

Exploring and implementing a single, client only, Android application.

Implementation of a basic backend application which will connect to DynamoDB via Lambda, and publish dummy notifications via SNS.

This phase will also include a simple Intent with few utterances and slots using Lex.

Design:

Basic UI/UX layout design.

Requirement design document.

Server Sequence Diagram.

Server Class Diagram.

Entity Data Model Diagram.

At the end of this phase, the candidate will deliver a simple functionality which will execute a command from the mobile via web method, store a trivial piece of data into the database, and receive a notification via the server's notification system whether the web method stored the data to the database successfully. In short, the deliverable of this phase will prove that all of the technology stack of this project functions, and prove readiness for real implementation.

<u>Developing and Programming: 9-12 weeks.</u>

Client – Development and deployment of the mobile application, as defined at the Interactive (section 5.2).

Server – Development and deployment for all of the Project's Entities using DynamoDB, along with business implementation via Lambda as a Web - services, and implementation for the SNS notification systems.

<u>Testing: 2 – 4 weeks.</u>

This phase will include functionality testing for each of the Lambda Web services, mobile usability testing for the entire client application's features. Integration testing will take place will multiple devices to simulate the interaction between multiple users. Deployment and testing will take place using "Staging" server configuration environment

Site Launch: 1 week.

This will include an initial release of the client application via Google Store, and deployment of "Production" server configuration environment.

7.Costs

AWS:

DynamoDB - 25 GB of Storage, 25 Units of Read Capacity and 25 Units of Write Capacity – enough to handle up to 200M requests per month with Amazon DynamoDB.

Lambda - First 1 million requests per month are free, $0.20 per 1 million requests thereafter ($0.0000002 per request)

SNS - First 1 million Amazon SNS requests per month are free, $0.50 per 1 million Amazon SNS requests thereafter.

Android Visual Studio: free for commercial use.

The initial estimation is that this project won't costs any money, or in worst case scenario, won't exceed few dozens of dollars per month. The tests and executions won't exceed million requests per month, as it is a school project. Also, it is unlikely that this application will exceed storage of 25 GB.

Bibliography

Sakr, S. (2013, April). To-do list app Remember The Milk gets a healthy update on Android.
https://www.engadget.com/2013/04/25/remember-the-milk-android-update.

Henry, A. (2013, May). Not Just Another Notes App: Why You Should Use Google Keep.
http://lifehacker.com/not-just-another-notes-app-why-you-should-use-google-k-509256637.

Google Android Developers, (2017). Introduction to Android
http://developer.android.com/guide/index.html.

Maxwell, E. (2017, January). The MVC, MVP, and MVVM Smackdown
https://news.realm.io/news/eric-maxwell-mvc-mvp-and-mvvm-on-android.

Butler, B. (2016, April). What is Amazon cloud's Lambda and why is it a big deal?
http://www.networkworld.com/article/3053111/cloud-computing/what-is-amazon-cloud-s-lambda-and-why-is-it-a-big-deal.html

Alexander, N. (2017, July). AWS Database Services Complete Overview: RDS vs Redshift vs DynamoDB vs SimpleDB, Cloudberry Lab. https://www.cloudberrylab.com/blog/aws-database-services-complete-overview-rds-vs-redshift-vs-dynamodb-vs-simpledb.

Lunden, I. (2013 August). Amazon's AWS Now Offers A Push Notification Service For iOS, Android And Kindle Apps, Widening Its Toolkit For Mobile Developers. https://techcrunch.com/2013/08/13/amazons-aws-now-offers-a-push-notification-service-for-ios-android-and-kindle-apps-widening-its-toolkit-for-mobile-developers.