James Clark
Tirumala Konireddy
Jagadish Tirumalasetty
Wang Zhang
Dr. Yugyung Lee
CS 560: KDM
February 25, 2014

Project Plan

Introduction

Group 6, self-named “Lee Students,” is a University of Missouri - Kansas City software engineering team training in big data knowledge discovery and management under the instruction of Dr. Yugyung Lee and her assistant, PhD candidate Feichen Shen.

Project Goal and Objectives

Motivation

We will apply data mining and machine learning tools and techniques to build an intelligent app for managing and organizing publicly issued safety notifications.

Objectives

The primary objective of this app is to aggregate public safety warnings and recall notices. By organizing safety data across categories, we aim to offer notification services to guests and subscribers. A guest should be able to search, sort, and read based on criteria they enter. Registered users should be able to subscribe and receive notifications based on their stored preferences.

Significance

A product safety app can prevent harmful losses relating to health, property, and finance. By collecting and processing a variety of public safety and recall notification sources, it is possible to provide useful consumer information in a single location.

Related Work

Recall Watch

This app is an aggregator of six data sources: FDA, CPSC, USDA/FSIS, MedWatch, Pet Health, and NHTSA. We have not considered using Pet Health or NHTSA. We may include them in the future. Further contrast between Recall Watch and our proposed design is the social media integration. Authentication occurs through Facebook, and users may share recall notices via

Twitter, SMS, and email. Reviewers of the app indicate detailed recall information is missing, navigation is cumbersome, and searching for information is not intuitive.

Recalled!

This app basically offers a reiteration of the detail pages linked to in the FDA and CPSC news feeds. We aim to offer additional data sources by category and provide a search feature. Furthermore, we would like to provide users with a login system and the ability to save preferences regarding recall notification. Reviews indicate performance is this app's best feature, but it lacks search functionality.

Recalls Plus

This app aggregates four data sources: CPSC, NHTSA, FDA, and USDA. This is perhaps the best app currently on the market, boasting reviews by Forbes.com, parenting.com, and the Apple App Store. The interface with its many images and large icons is visually appealing. Features include authentication, storing preferences, registering products, and receiving notifications. On its downside, Recalls Plus offers fewer data sources than Recall Watch. Reviewers indicate performance may be an issue and software bugs are common. There may be some problem related to local storage as a couple users are asking for the ability to move data onto an SD card as the app is eventually filling up their available storage.

Proposed System

Requirement Specification

Functional Requirements

Default categories should be provided to the user for easy sorting and searching. The interface should be visually comprehensible and intuitively navigable by the user while at the same time comfortable within the screen size constraint of a small Android smart phone. Users should be able to login, filter results, save preferences, and subscribe to receive notifications.

Non-functional Requirements

Operational

Using minimum Android SDK 11 Honeycomb should simplify development. Further simplification can be achieved by designing any web interfaces to support a minimum of HTML5, and CSS3.

Performance

We expect to provide the best mobile application performance possible by implementing a native Android application with limited webview elements. The application is ultimately a parser of results produced by a hadoop/mahout/solr application. The parsing of data should be very quick as the heavy processing will be previously accomplished through server-side bashing.

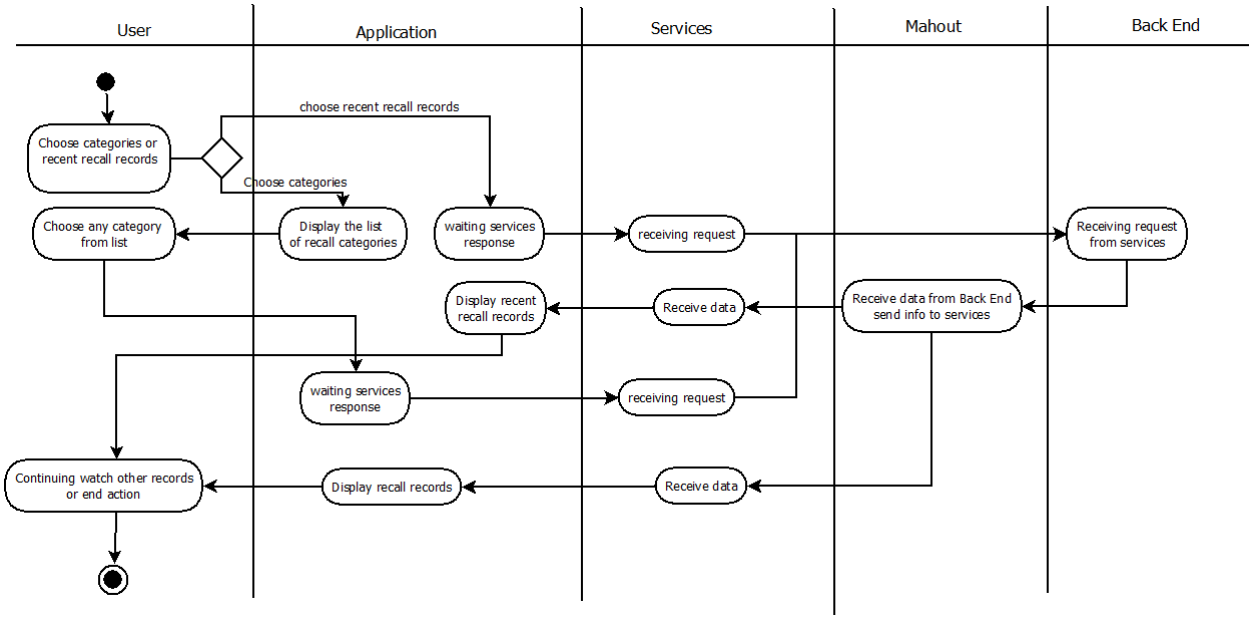
Security

It is possible we will use three storage locations. First, the processed data results at the server are publicly available through RESTful web service. Second, any data storage we conduct with our users are beyond the scope of this course and require additional technology not included with instruction. For example, we can store preferences in MongoDB servers on Heroku's cloud platform, or write web services to handle data through Google App Engine's High Replication Datastore service. Third, local data can be stored on individual users' devices whereupon we will investigate the type of data we may need to store and the varying degrees of associated security.

Business / Technical Requirements

A server will need to cycle a batch process manager task to repeatedly poll data sources for updates

Business Process/Workflow analysis (UML Activity Diagram)



Framework Specification

System Architecture Diagram

Domain Model

The finally converted structured data is stored in the server in the form of a json table as below. In this table, one dynamic column is allotted for every category. The category name or id and alerts were stored in the first rows of every column respectively. One dynamic row is created for every user in which the first column contains the username or id, and parity bits '1' or 'o'. ('1' under subscribed category column and '0' under unsubscribed category or '0' is the default parity for every category for user unless user is subscribed to that category. If a query is received by server to retrieve a particular user alerts, it first searches in the first column for username or id

and then send the alerts stored in the second row for the subscribed categories(categories whose parity bits are '1' in the username row) or search category.

Fig. 1: Json array table

Category	Climate	Honda	Windows 8
Alert	1. Today Temperature 15^0 C. 2. Showers from 06.00AM to 10.00AM. 3. Severe Thunderstorm alert in evening.	1. Speed limit 35mph in today's snow. 2. Park your car in some hide.	1. Upgrade your software Windows 8.1. 2. New features were added to ms word.
James	1	1	1
Tiru	1	0	1
Wang	1	1	1
Jagadish	0	1	0

Methodologies

Front End Methodology

1. Get User credentials from user.
2. Send a query about the requested alerts through webservice to data-base server.
3. Retrieve the whole json array of alerts using hadoop file system that were stored in the server under that user credentials.
4. Display those alerts on the user home screen and now user can see all his alerts.
5. If user request any new category alerts through search, the app sends a query of 'search text' to server through Solr to analyze search text and then hadoop file system to is used to search the category in the server to retrieve that subscription array.
6. If user wants to subscribe that service, then that service is appended to the user's json array in server using mahout and from then the that service alerts were synced with the recent recall alerts that were shown in home screen.

Back End Methodology

1. Our system keeps on updating the rss feeds and websites data document(google document) every day for any appended data to them.

2. If any new data or information is detected, our system stores that data into a temporary dynamic variable.
3. Run adaptive decision tree using mahout where we use Case base reasoning and naïve bayes classifier to check whether there is any relevant information to alert or not.
4. If the decision tree returns a alert of some category, store that alert in server of json array.

Algorithms

It makes sense to begin work using a naive Bayes classifier while continuing research in new algorithms and systems that improve relevance.

Analytic Tools

1. Mahouth: For implementing Adaptive decision tree, Case base reasoning and naïve bayes classifier.
2. Hadoop: For using distributed system to run Mapping, reducing functions and store data into server(json table).
3. Solr: To map user entered search text to relevant category while searching and retrieving data from server. We may use it for analyzing data from rss feeds and html files.

Analytical Tasks

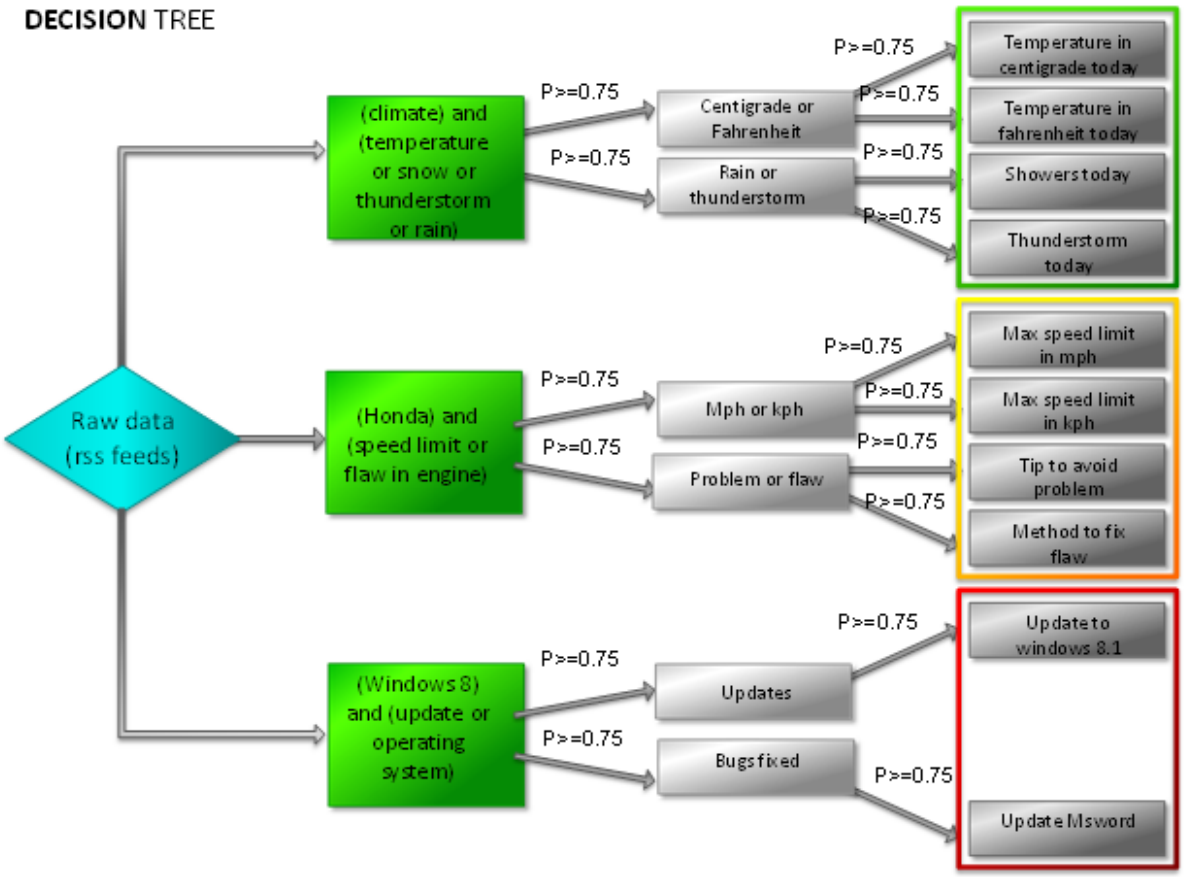
Front End Tasks

1. Query to server using Webservices about the user alerts using his credentials.
2. Display and sync the alerts in home screen with newly subscribed alerts or updated alerts.

Backend Tasks

1. **Case boundary system**: In case boundary system we define possible cases of the data which notifies an alert. It may be the combination of matches or some Boolean algebra logic between combination words. Each case is tested using naïve bayes classifier's probability.
2. **Naïve bayes classifier**: for calculating the probability or the factor of case satisfied by raw data.
3. **Adaptive decision tree**: We use adaptive tree for analyzing the raw data to convert into structured data. Here, we use the case boundary system at every node to analyze the information. While. Training we use some news feeds containing alerts to find the common combinations of words in the alert containing news feed. Based on the training, we fix the factor of correlation or matching required for the case in every node to get satisfied for raw data of the decision tree.

Fig 2: Decision Tree:



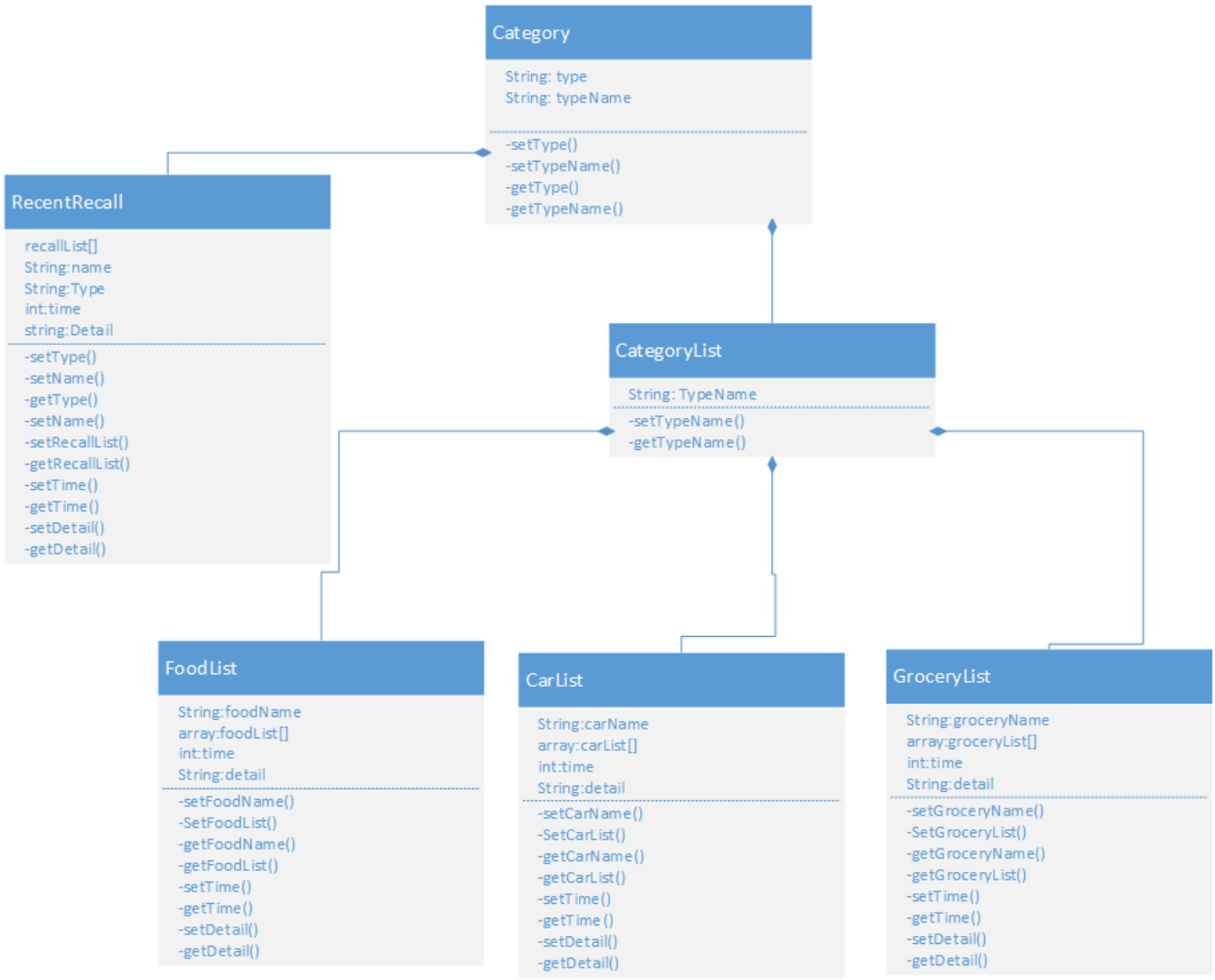
Source: Decision tree model using case boundary system.

System Specification

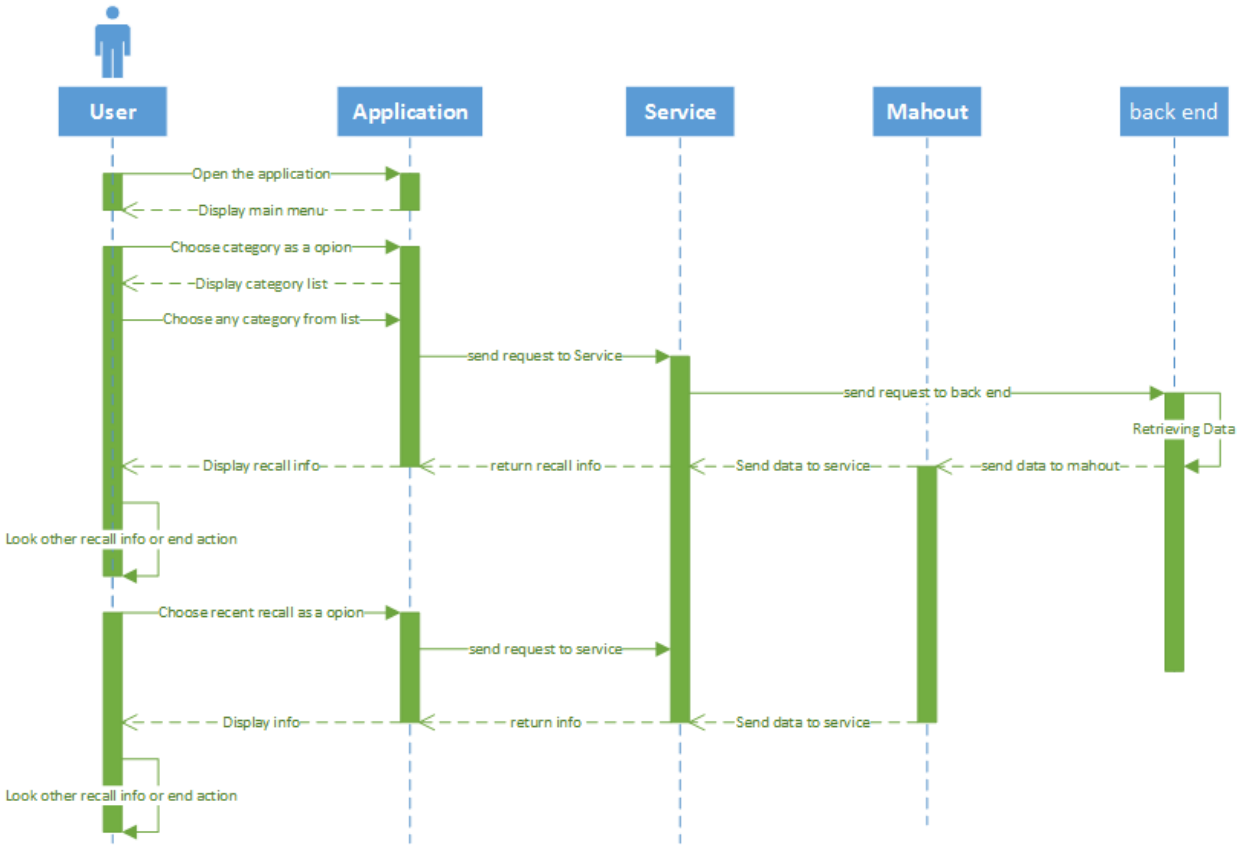
Existing Services:

Consumer Product Safety Commission. The sources come from consumer product safety commission government website, it contain all kinds of categories recall records.
http://www.cpsc.gov/en/Newsroom/CPSC-RSS-Feed/Recalls-RSS/
General Recalls. The sources come from USA.gov website, it contain child satety seat recalls, food recalls, drug recalls, tire recalls and some other recalls information.
http://www.usa.gov/Topics/Reference-Shelf/Libraries/RSS-Library/Consumer.shtml#Recalls
Food and Drug Recalls. The source come from US Food and Drug Administration website, it contain food and drug recalls.
http://www.fda.gov/AboutFDA/ContactFDA/StayInformed/RSSFeeds/Recalls/rss.xml
Consumer Product Safety Information Database. The source come from saferproducts government website.
http://www.saferproducts.gov/WebApi/Cpsc.Cpsrms.Web.Api.svc/

Class diagram



Sequence diagram



Service Specification

Base on the services (CPSC, NHTSA, FDA, and USDA), we will store these data as XML or Jason format, then we will extract recall information from services, such as the name of recall items, the date of recall items, the reason of item recall and description of recall items. as output we will display exact recall information.

Design of Mobile Client

Our project Application is Android based. For our app GUI, it contains two main feature. One is recall categories, in this part, we will separate each category base on our data for user convenience. Another is recent recall records. In this part, our App will display most recent recall records. Further more, we will add a search bar to enable user search whatever they are looking for. Also we may implement a user login function and registration function.

Plan by Services (using ScrumDo)

Schedule for the four different increments (for each increment, do the following tasks)

- Epic: Increment 1

        - track RSS feeds
            - Tirumala: Google Spreadsheet as data source
                - note: probably easier to just download the RSS as xml file and include in project
        - apply machine learning algorithm to all feeds
        - develop GUI
            - include pre-sorted categories
- Epic: Increment 2
    - refine machine learning algorithm
    - Create web services to fetch data from Data store to the Web UI application
    - add search feature on top of pre-sorted categories
    - Implement all functionalities involving data connections (retrieving items list on the fly, passing required parameters to all necessary services) and develop all Data Services.
- Epic: Increment 3
    - authentication and social media sharing
    - refine GUI
- Epic: Increment 4 - Refine GUI

- Epic: Final Project - Deploy the components and complete documentation.

Stories (features): Scenario & Use case specification template

Project Timelines, Members, Task Responsibility

Bibliography

Recall Watch. Computer Software. Google Play Store. Vers. 2.0.1. Stellar Computer Systems, 11 Jul. 2011. Web. 1 Jan. 2014. <https://play.google.com/store/apps/details?id=com.stellarpc.feeds.recallwatch&hl=en>

Mack, R. *Recalled!* Computer Software. Google Play Store. Vers. 1.03. Mack, R. 6 Jun. 2011. Web. 1 Jan. 2014. <https://play.google.com/store/apps/details?id=com.rmackconsulting.recalled&hl=en>

Recalls Plus. Computer Software. Google Play Store. Vers. 1.1.1. SAP AG. 27 Apr. 2012. Web. 1 Jan. 2014 <https://play.google.com/store/apps/details?id=com.sap.recallsplus&hl=en>