STORE SELECTOR

Group 06

December 2012

Team: Blair Billings
Timothy Kalpin
Kurt Kohl
Chris Morgan
Kerrick Staley

Client: Google
Advisor: Manimaran Govindarasu

Version	Editor	Date	Peer Reviewers	Notes
0.1	Blair Billings	February 13, 2012	Chris Morgan	Draft 1
0.2	Kurt Kohl	March 30, 2012	Chris Morgan	Draft 2
0.3	Tim Kaplin	April 17, 2012	Kurt Kohl	Final Draft

Table of Contents

Table of Contents

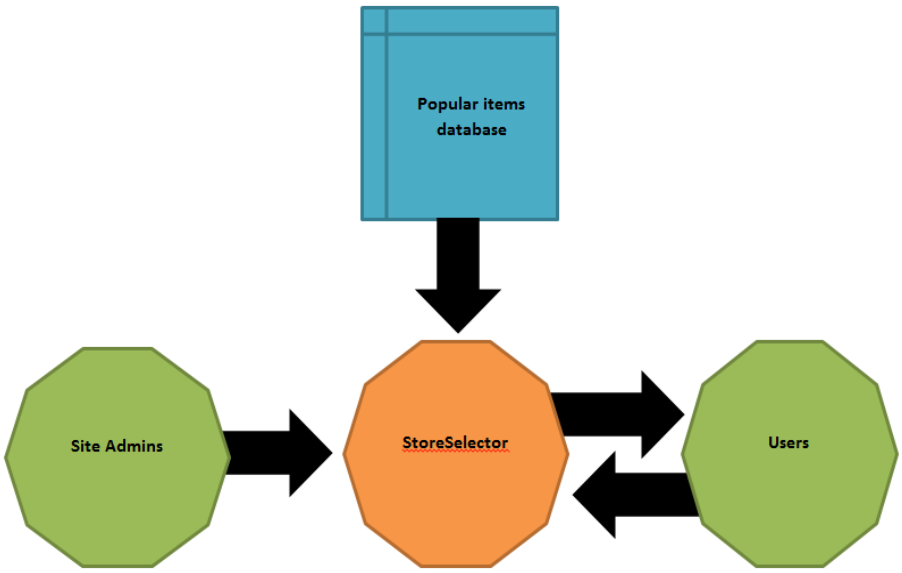
➢Section 1 - Introduction

➢1.1 Problem-Need Statement

There are a lot of different places people can go to gather data about deals and pricing information for local shopping items. However, this data is very widespread, and the searches required have to be repeated on a weekly (or daily, monthly, etc.) basis. Often such deals are also relatively inconvenient especially in the case of local shopping, as most deals require presenting some coupon or code at the time of purchase.

Therefore, it is necessary to create a single public resource to query these shopping items and remember each users' searched items.

➢1.2 Concept Sketch



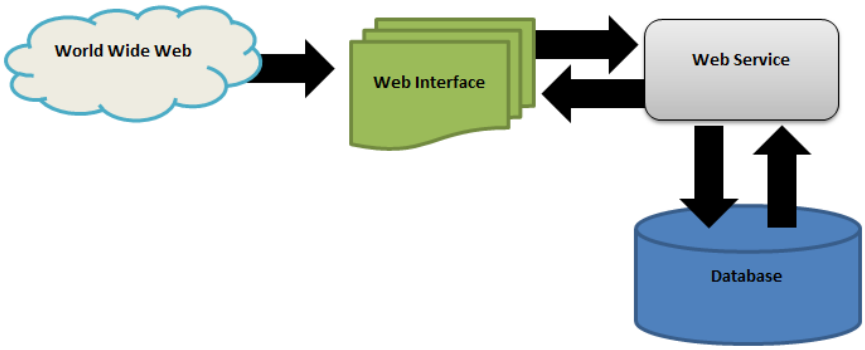
➢Section 2 - System Description

➢2.1 - System Components

The system shall consist of the following components:

- The database, which shall store information about product prices as well as user account data.
- The web backend, which shall query the database upon incoming user requests and generate HTML pages in response.
- The web frontend, which shall allow users to easily access the site's functionality through either mobile or traditional (desktop) web browsers.

➢2.2 - System Block Diagram



➢Section 3 - Operating Environment

The system will require a host server to house the application. This could be either a physical machine, or a virtual server hosted on some other platform. It should be consistently reachable, and will require either a static IP address, or a DNS to ensure its availability. The server must also be able to make the resources we plan to implement available to mobile users. Because of this, it should not require the client platform to perform much computation, if any. Also, because it will make use of client information, the host must ensure some degree of security.

Because this service is intended for use by a large number of users it must be scalable and able to handle not only a large number of connections, but also concurrent updates from several users.

➢Section 4 - User Interface Description

The system shall serve two separate types of end-user:
Consumers: The interface that is presented to consumers shall allow them to easily select the products they're interested in from among the entries in the database. After the consumer has selected a list, it shall calculate the amount of money that can be saved by shopping at each nearby store, and display this information on a map for the user to view. The software shall also make suggestions for similar but less expensive products as well as track which products the user likes so that they may more easily construct shopping lists in the future.
Sellers: The seller interface shall allow store managers to easily enter and update pricing information for many products at once and ensure that the price entries are correctly matched to the product entries in the database. It shall allow this data to be uploaded in common formats such as .csv, .odt, and .xls; it will accommodate the sellers' existing computer systems, rather than requiring those systems to accommodate it.

➢Section 5 - Requirements

➢5.1 - Functional Requirements

1. The system shall allow users (e.g. shoppers) to find the total price of items they are seeking, using their Android device.
    a. The system shall accept a product name via voice and/or keyboard input.
    b. The system shall match this input to its database of known products.
    c. If there are multiple matching products, the system shall present a list for the user to select from.
        i. For keyboard input, this list will be generated as the user types (i.e. it will autocomplete the input).
    d. The system shall present a means of adjusting item quantity.
    e. The system shall build an on-screen list of items during this process.
    f. The list shall allow deletion of entered items.
    g. Once the user has completed the list, the system shall calculate the total price of the items at various nearby stores, and display this information on a map.
    h. The system shall allow saving, opening, and modification of product lists.
2. The system shall allow users (e.g. store managers) to submit price information for their products via a web interface.
    a. The system shall allow users to upload a price table in .odt, .xls, or .csv format, or to link to a Google Doc containing price information.
    b. The system shall allow users to indicate which columns in the table indicate the name and price of each product.
    c. The system shall allow users to view, search, and edit uploaded data.

➢5.2 - Non-Functional Requirements

1. The user interface shall be easy to use and aesthetically pleasing.
2. The backend shall process 120 queries per minute, and be able to scale to process 10,000 queries per minute with sufficient hardware.
3. The code in the final product shall be modular, readable, and well-documented.
4. System shall be completed by end of Fall Semester (December) 2012

➢5.3 - Security Requirements

1. System shall store user login information (passwords) in a secure manner (using encryption).
2. System shall require authentication to access user data.
3. System shall not serve user data to other users.

➢Section 6 - Market and Literature Survey

There are several Internet sites out there today which offer pieces of what Store Selector will provide, but each of them has gaps that this application will fill. Websites like shoplocal.com have deals that you can search for, but there are no accounts, so a user would have to repeat their searches over and over each time they wanted to search for deals. Plus, they do not have a mobile application. Applications like LivingSocial, Groupon, and ScoutMob offer account based web-applications, but they are for larger cities. So, it you don't live in or around Des Moines, as an Iowan your town's deals could be left out. There also seem to be applications such as GeoQpons and The Coupon App that will give you only coupons, which is not the sole purpose of our application. And, apps like ShopSavvy will only give you prices, again demonstrating the need for these features to come together.

The major differences of our application are as follows: our application will be more local than some of the big-market services. Also, our application will be a one-stop location for both regular prices and deals at places you frequently or infrequently shop.

➢Section 7 - Deliverables

- Database (for storing User data)
- Web Interface
- Mobile Interface
- Voice Interface (for inputting shopping list)
- Web Entry Page/Email scrubber

Our application will provide a web interface and with a back-end database to store information about our users and their shopping lists. The shopping lists will be given as a void input or text input and we will provide a web page and email scrubber for deal entry so users can search for pricing and deals associated with items that match their search criteria.

➢Section 8 - Work Plan

➢8.1 - Work Breakdown

We decided to divide the work by deliverable, as follows.

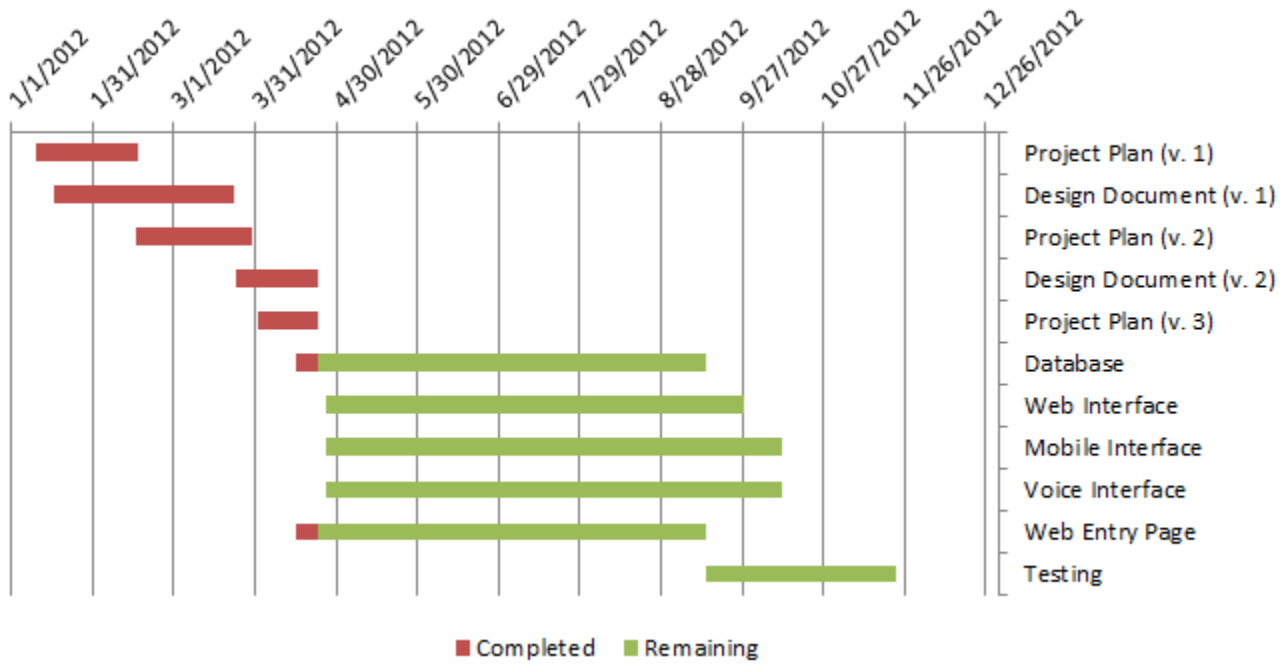
- Database - Chris
- Web Interface - Blair

- Mobile Interface - Kurt
- Voice Interface (for inputting shopping list) - Timothy
- Web Entry Page/Email scrubber - Kerrick

➢8.2 - Resource Requirements

This project will require a server as described in the operating environment section, as well as a domain name. We may possibly consider a site certificate in the future to ensure user data stays secure by helping prevent phishing efforts.

➢8.3 - Project Schedule



➢8.4 - Risks

- We have not been able to fully formalize a set of requirements with our client, so the requirements could change drastically in the future. We will therefore use an agile-development methodology, which will allow us to more rapidly react to changing customer demands.
- It is unclear how much pricing information is available on the web; we may not be able to provide end users with enough information to make accurate decisions. We will therefore allow users to flag inaccurate prices and supply more accurate price values. We will also encourage companies to submit their pricing information.

➢Section 9 - Testing

9.1 Testing Requirements

The obvious goal of any software testing suite is to discover and report as many bugs as possible in the software. For the back end of the software these can be largely automated tests to simply ensure that the system works as desired under any input while testing for edge cases such as blank fields, illegal characters, and oversized random strings. Also, stress testing for the backend is more or less impossible without some form of automation. Because this kind of testing is so prevalent there are many options available to generate and deploy tests of this nature.

For the front end, the system will undergo functionality testing. This problem is less easily solved by automated testing, and will require user feedback. Once the project's backend is up and running, we may release the software in a limited manner for alpha and later beta testing. This will allow us to receive user feedback based on how they want the system to act, and how easy they feel it is to interact with the system. This will also present an opportunity for more bug reporting throughout the system.

9.2 Testing Breakdown

9.2.1 Front End Testing

Our application will undergo strenuous usability testing on the frontend. As a whole our application needs to be functional, but still very easy to use. Have usability tests will help us gauge our progress on that front. Given the opportunity to log in, we will have test users try to perform specific, necessary functions. They will then report how they feel the process went, and we will observe their testing as well.

9.2.2 Back End Testing

The largest concern with the back end of our application is stress testing. We need to see if our application can handle many users all writing to, or requesting information from our database simultaneously. We will also use fuzzing to give our application random and incorrect data to see how it handles that data. This will also show us if any random data could crash our application.

➢Section 10 - Technology Considerations

We assume that all users will have either a mobile device (like an Android phone) or a desktop computer with which to use our product. As this is a Google project, we will be targeting mobile platforms, making the Store Selector lend itself to its mobile applications, but especially those with an Android operating system.

➢Section 11 - Intellectual Property Concerns

This is currently an academic project, but may eventually be used by Google. This raises concerns as to which software systems are available. MySQL is available for use as a database, but is currently released under a FOSS license, and may at some point require availability of the source for this project if it were ever deployed by Google. This is not the only part of the software in which this issue arises, but shows the important issue. This will of course be alleviated if Google substitutes its own software for the libraries we choose to use. At which point the FOSS license will become a non-issue.

➢Section 12 - Closure Material

12.1 Project Team Information

12.1.1 Client

Google
Muthu Muthusrinivasan
muthup@google.com

12.1.2 Advisor

Manimaran Govindarasu
3227 Coover
Ames, IA 50011-3060
515-294-9175 (Office)
gmani@iastate.edu

12.1.3 Team Members

Blair Billings
Computer Engineering
bbill7@iastate.edu

Timothy Kalpin
Computer Engineering
trkalpin@iastate.edu

Kurt Kohl
Computer Engineering
kdkohl@iastate.edu

Chris Morgan
Computer Engineering
cwmorgan@iastate.edu

Kerrick Staley
Computer Engineering
kerrick@iastate.edu