Yoon's Whiteboard:

Applying Mobile and Internet Technology to Developing Regions

Yoon-Sung Hong
Matthew Lauren
Ting-Yen Wang
CSE454, Fall 2010-2011

Introduction and Goals

In developing regions, the Internet infrastructure is lacking, but the mobile infrastructure is quite well established. The Open-Data-Kit (ODK) research group at UW found that SMS transmission time in Tanzania is faster than that in Seattle. ICTD and Mobile research communities have been investigating how to utilize the mobile infrastructure in developing regions to aid healthcare, economical, and social problems. The research communities and ODK group inspired our group to design and develop an information system that fits within the developing region context by taking advantage of the established mobile infrastructure. Particularly in this class, our group focused on improving the supply chain of the farming industry in developing countries by providing a way for farmers to reach out a broader potential consumer base with less middlemen through mobile and Internet technologies. We believe that our system can help farmers make fair deals and posses a larger portion of the profits by slimming down the supply chain. Designing such a system requires adopting a collection of technologies. We intended to learn how to choose the appropriate technologies for the system and create a harmonious and effective system. Also, due to the heterogeneous nature of our system design, it was our goal to figure out the most effective communication protocols between the components of the system. On the implementation aspect, we intended to learn cloud computing platform and Hadoop. Finally, we planned user studies to verify that our implementation is usable by people with diverse technical backgrounds. We believe that these user studies can give us some ideas as to how people in developing regions will accept our implementation.

System Design and Algorithm

Overall Architecture of the System

The following sections will describe the implementation of each of the components of our system, Yoon's Whiteboard. Figure 1 provides a high-level overview of each of the components and how they interact with each other.

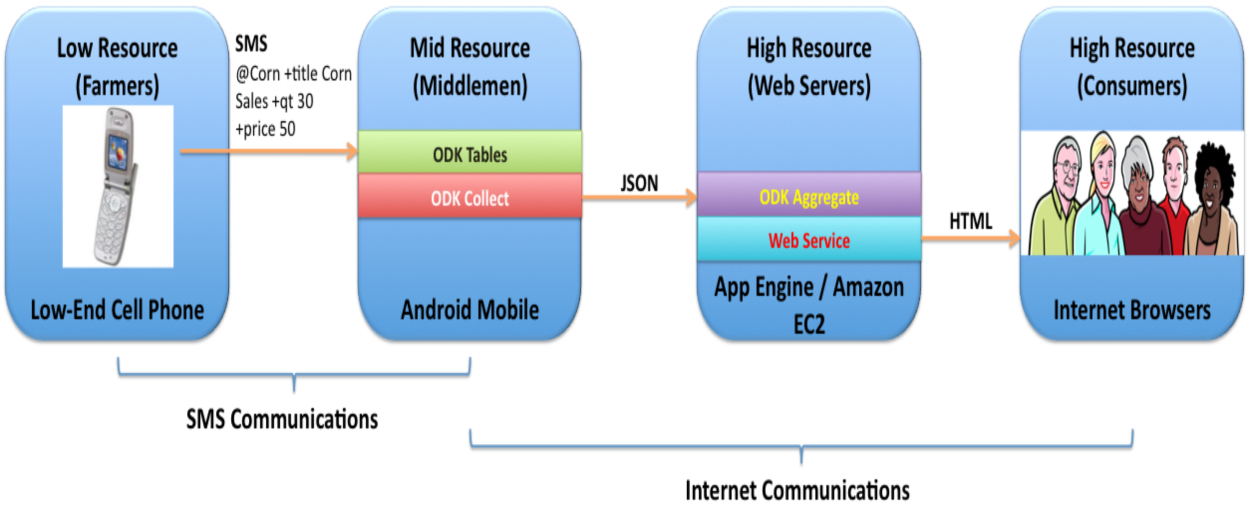
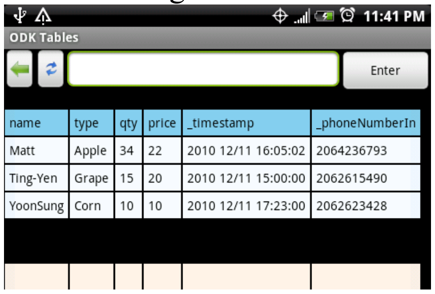


Figure 1: High-level view depicting overall design of Yoon's Whiteboard

Components

ODK Tables

ODK Tables is an application similar to Excel built on the Android mobile platform to help manage a collection of SMS messages (see Figure 2 for screen shot). You can update and retrieve information to the application running on Android mobile device through SMS.



name	type	qty	price	_timestamp	_phoneNumberIn
Matt	Apple	34	22	2010 12/11 16:05:02	2064236793
Ting-Yen	Grape	15	20	2010 12/11 15:00:00	2062615490
YoonSung	Corn	10	10	2010 12/11 17:23:00	2062623428

Figure 2: Screen shot of ODK Tables

ODK Collect

ODK Collect renders an Xform into as sequence of input prompts that apply form logic, entry constraints, and repeating sub-structures (see Figure 3 for screen shot). Users work through the prompts and save the submission as completed or partially completed (allowing later revision). ODK supports GPS and the various methods of input such as video, audio, and photos on Android mobile platforms.

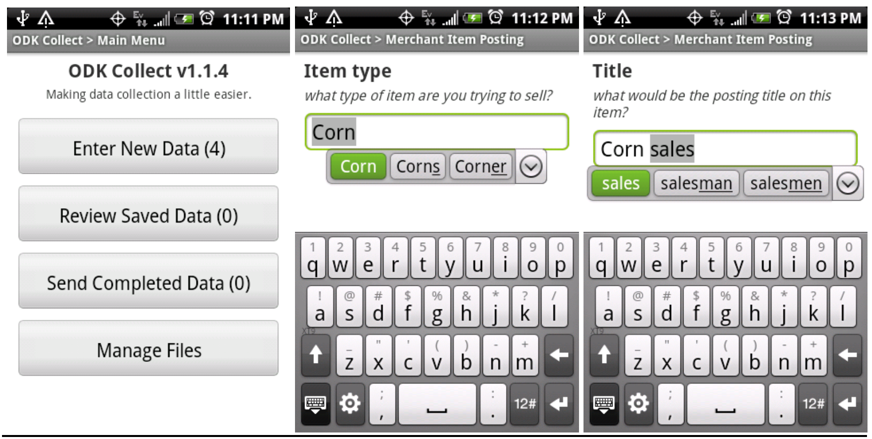

ODK Collect > Main Menu
ODK Collect v1.1.4
Making data collection a little easier.
Enter New Data (4)
Review Saved Data (0)
Send Completed Data (0)
Manage Files
ODK Collect > Merchant Item Posting
Item type
what type of item are you trying to sell?
Corn
ODK Collect > Merchant Item Posting
Title
what would be the posting title on this item?
Corn sales

Figure 3: Screen shot of ODK Collect

ODK Aggregate

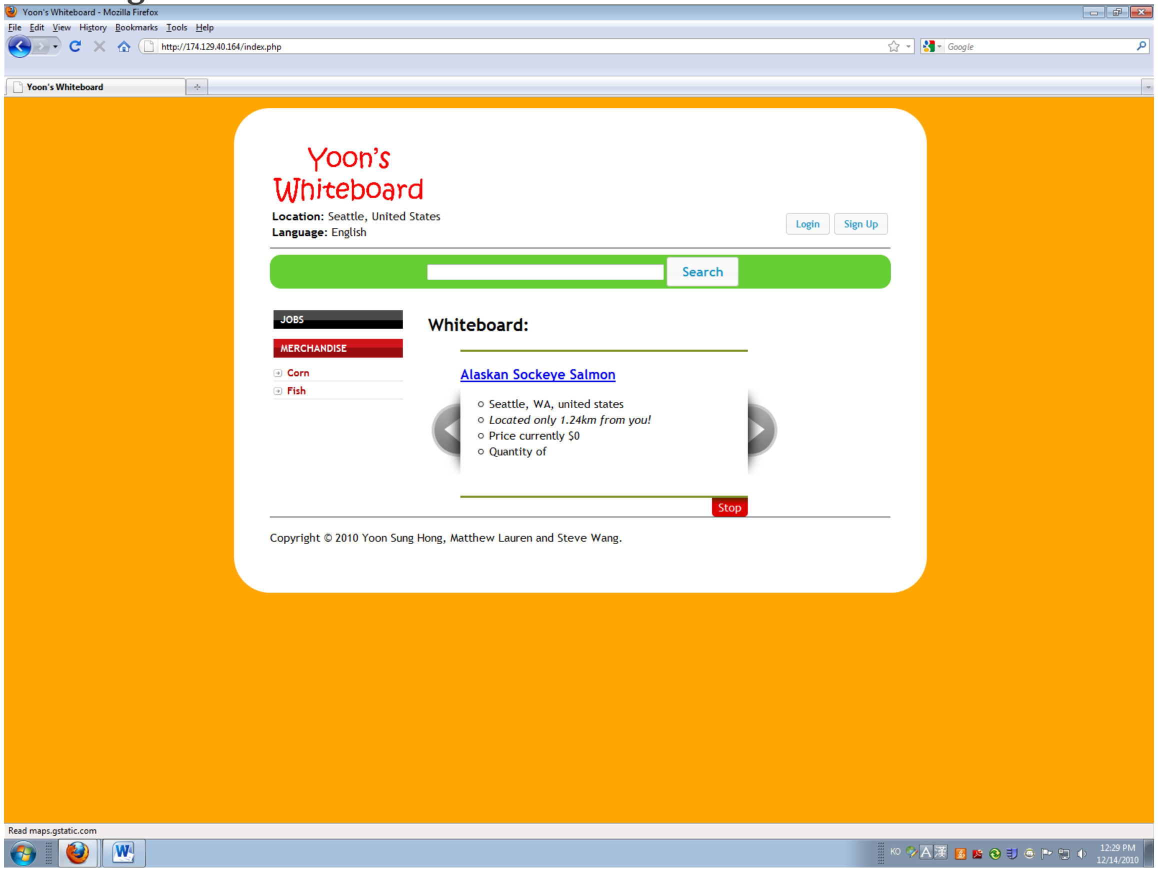
ODK Aggregate is the server pair for ODK Collect. ODK Collect sends data in JSON format and Aggregate instances that run on the Google Engine forward the data to configured web servers in JSON format.

Website

Yoon's Whiteboard is similar to Craigslist, but designed for developing regions. Users can sell and buy products through the website. Users can make postings to our website either through web browsers or ODK Collect on a mobile device. The website is built with the LAMP architecture where we use Ubuntu Linux as our platform, Apache as our web server, MySQL as our database engine, and PHP as our back-end and front-end programming language. The site is housed on Amazon Web Services (AWS) on a small EC2 instance – a small EC2 instance is defined as having 1 virtual core with 1 EC2 Compute Unit (equivalent to 1GHz), 1.7GB of memory, 160GB of instance storage, a 32-bit platform, and moderate I/O performance.

Continues on Next Page Regarding Website Features

1. Main Page



Location: Seattle, United States
Language: English

Alaskan Sockeye Salmon

- Seattle, WA, united states
- *Located only 1.24km from you!*
- Price currently $0
- Quantity of

Copyright © 2010 Yoon Sung Hong, Matthew Lauren and Steve Wang.

Figure 4: Main Page of Yoon's Whiteboard

Location/Language

Our service is community oriented. Capturing the location of the user is essential in many parts of the website, including search results and suggestions. Capturing the user's location is performed by detecting the user's IP address and then translating that IP into a latitude, longitude and region code – such as city, state, country; this translation of IP to location is done through a library call that we implemented that requests the mappings from an online database called IPInfoDB. We prominently display the user's region code to the user in the header (see Figure 4) and utilize the latitude and longitude to help user's identify nearby products in their community. In addition to location, we designed our web service to also serve many regions. Capturing and providing the user's appropriate language are essential for the future implementations of the system.

Search Bar

Search bar is the easiest way to perform general purpose searches on the website. As Figure 4 demonstrates, we tried to make the search bar easy to find by adding color to that portion of the website. Figure 4 also demonstrates our desire for ease when searching by keeping the interface simple. More details concerning how the search bar works are below in "Search."

Menu

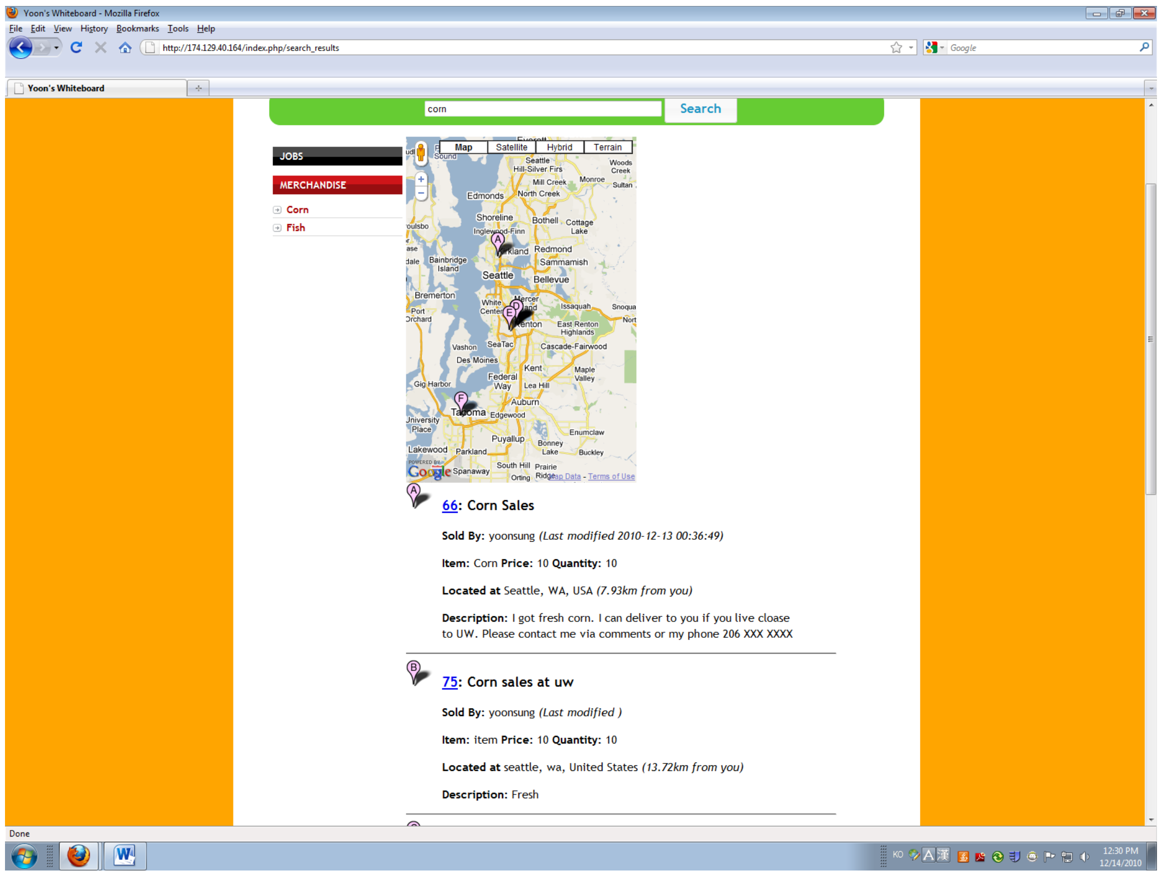
- **Merchandise**
  Provides quick links to popular categories such as corn and fish (see Figure 4). By clicking the links, users can see all listings relevant to the category. The "Merchandise" aspect of our website has been fully implemented.
- **Jobs**
  A community oriented website can be a very good space to seek or advertise jobs. This is especially true in developing regions, as most jobs in this context are hired locally. Note that the "Jobs" portion of the website is not implemented since our focus in this class was to improve the supply chain of the farming industry. This section exists to show the potential extensibilities of the web service.
- **Slider of Suggested Products**
  On the front page, customers are immediately presented with items that may potentially interest them (see Figure 4). Our focus on community and location is demonstrated here in that the current design lists the top seven closest items to the user. As mentioned above, the user's location is detected by translating the user's IP address into a latitude and longitude; as we will discuss in more detail in the "Making a New Posting" section, all items are also tagged with latitude and longitude attributes; this item and user location information allows us to identify those items closest to the user, which may provide the greatest allure to the customer. When the website begins to receive more traffic and we are able to conduct some customer profiling, the suggestions could be expanded to include the parameter of customer buying habits.

2. Admin and Access Control

To fully utilize our website, users need to login. We have fully implemented user authentication system in a very secure way using encryption techniques such as MD5 + Salt. User studies shows that more than 90% of participants had no trouble signing up, logging in and managing their accounts on our website.

Continues on Next Page Regarding Website Features

3. Search


Yoon's Whiteboard - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://174.129.40.164/index.php/search_results
Yoon's Whiteboard
corn
Search
JOBS
MERCHANDISE
Corn
Fish
Map
Satellite
Hybrid
Terrain
Done

66: Corn Sales

Sold By: yoonsung *(Last modified 2010-12-13 00:36:49)*

Item: Corn **Price:** 10 **Quantity:** 10

Located at Seattle, WA, USA *(7.93km from you)*

Description: I got fresh corn. I can deliver to you if you live closae to UW. Please contact me via comments or my phone 206 XXX XXXX

75: Corn sales at uw

Sold By: yoonsung *(Last modified)*

Item: item **Price:** 10 **Quantity:** 10

Located at seattle, wa, United States *(13.72km from you)*

Description: Fresh

Figure 5: Search Results Shown with Google Map

The community orientation of our service led us to invest a great deal of effort into the design of a unique way of searching that is informed by the customer's location and his or her proximity from desired products.

Listing

After the user types in a search string depicting products the user is interested in, we provide a list of top product hits that are each accompanied with item details like the seller name, price, description, and the distance between the product and the user (see Figure 5). Like the main page's "Slider of Suggested Products," one of the most influential parameters of search is this distance parameter, as our original goal was to somehow sort our search results based on distance. However, differing from the product slider and its sole focus on distance values, search has to deal with the complexity of incorporating distance into the existing methods of searching for products through text search. We accomplished this goal of mixing distance with text search by utilizing a number of tools and algorithms, one of the most significant of which is MySQL's natural language full-text search capability with the FULLTEXT index. First, after the user enters a search string, we utilize MySQL's FULLTEXT index to parse the string and compute the relevance score of each product against the search string – the scores are floating point numbers from zero on up, where a zero means the product has zero relevance to the search string. The relevancy scores are computed similarly to the Naïve Bayes algorithm in that the number of occurrences of a word in the database signifies its significance towards the search string – the greater the number of occurrences, the lower the significance. In general, the greater the number of 'significant' words in the product listing that matches the search string words, the greater the relevancy score. The FULLTEXT index also incorporates the filtering of stopwords, giving them zero semantic value. We return those products with relevancy greater than zero as our initial list of search results.

In addition to those products that have been calculated as having greater than zero relevancy, we also add to the search results those products where the prefix matches the search string, since people often search for items by typing the first few characters of the product string. An example of a prefix search is providing "Salm" as the search string in order to find the product "Salmon Fillets." Since prefix searching can use non-words, this can often help the user find items that would have resulted in zero relevancy through the FULLTEXT index.

Once we have our list of search results, we need to re-score each of the products so as to create a listing order to present to the user. It is here that we need to start incorporating location and distance into the search results. As we will describe in the "Make a New Posting" section below, each product is tagged with a latitude and longitude describing where it is located. As mentioned above in the "Main Page" section, we also tag the customer's location (latitude and longitude) through the customer's IP address. Thus, with the latitude and longitude information for products and customer, we can utilize the Haversine formula to calculate the distance between the customer and each of the products in our search results – the Haversine formula calculates the distance in kilometers between two points on a globe and is sometimes known as the "as the crow flies" distance.

Finally, we add 0.04 to each of the relevancy scores and divide each of the product distances by their updated relevancy scores to finally get our listing-order scores (see Figure 6) – those products with lower listing-order scores are listed higher in the search results. We add 0.04 to each of the relevancy scores, since many prefix search results have a relevancy score of zero; the 0.04 padding provides these prefix search results with a 1 km to 25 km ratio such that if the product is close by and only 1 km away from the user, then the penalty for having a zero relevancy is only a factor of 25 km.

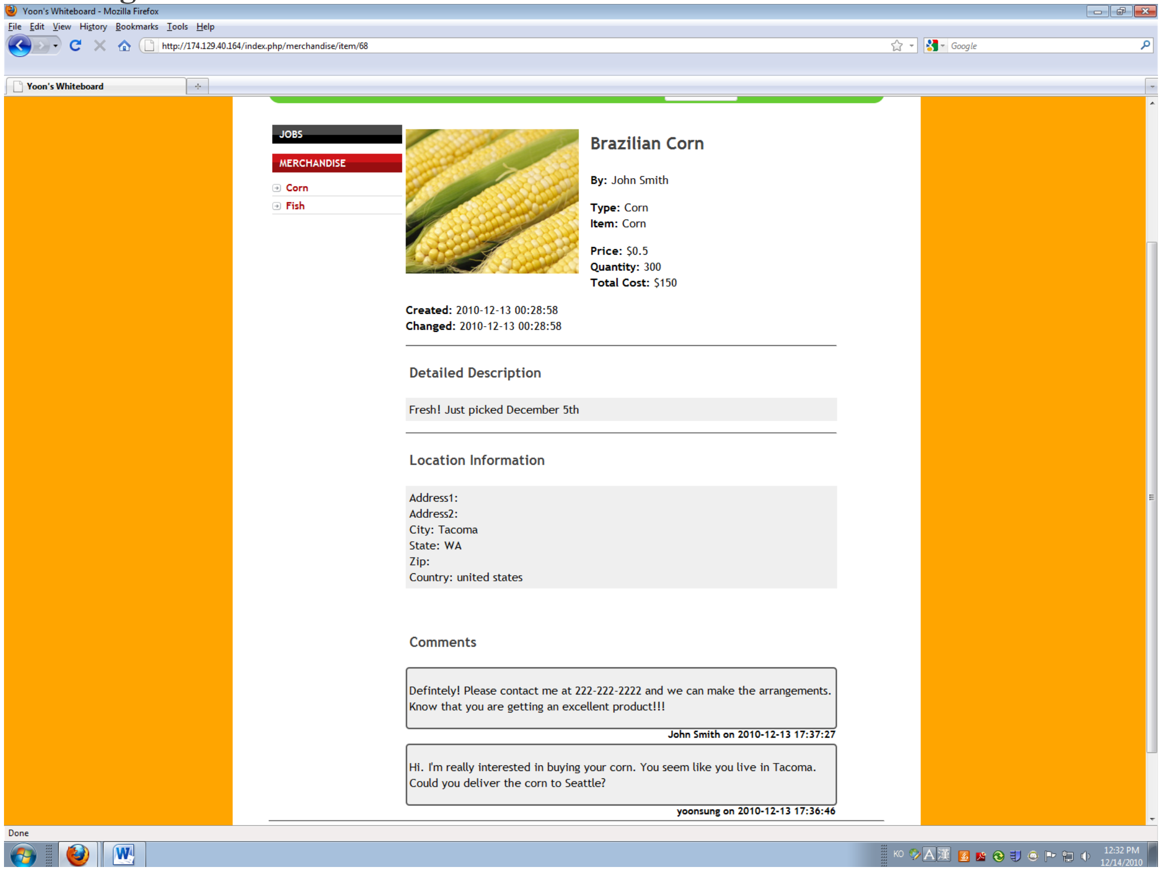
Listing Order Score = (distance between product and user) / (relevancy score + 0.04*)*

Fig 6. Search listing order score equation

Map

In addition to ordering each of the products with respect to both full-text natural language relevancy and distance from the user, we also reinforce the community-centered focus of search by displaying a local map of the products that are close to the user (see Figure 5). The map is generated with the Google Maps API. The map is centered on the user's location, as identified by the user's IP address, and the products are marked on the map with annotated markers placed on the map ("A", "B", "C", etc). Each of the search result listings is prefixed with an equivalent annotated marker, making it easy to associate the markers on the map with the associated search result listings.

4. Item Page & Comments


Yoon's Whiteboard - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://174.129.40.164/index.php/merchandise/item/68
Yoon's Whiteboard
JOBS
MERCHANDISE
Corn
Fish
Done

Brazilian Corn

By: John Smith

Type: Corn
Item: Corn

Price: $0.5
Quantity: 300
Total Cost: $150

Created: 2010-12-13 00:28:58
Changed: 2010-12-13 00:28:58

Detailed Description

Fresh! Just picked December 5th

Location Information

Address1:
Address2:
City: Tacoma
State: WA
Zip:
Country: united states

Comments

Definitely! Please contact me at 222-222-2222 and we can make the arrangements. Know that you are getting an excellent product!!!

John Smith on 2010-12-13 17:37:27

Hi. I'm really interested in buying your corn. You seem like you live in Tacoma. Could you deliver the corn to Seattle?

yoonsung on 2010-12-13 17:36:46

Figure 7: Example of Item Page for Merchandise Category

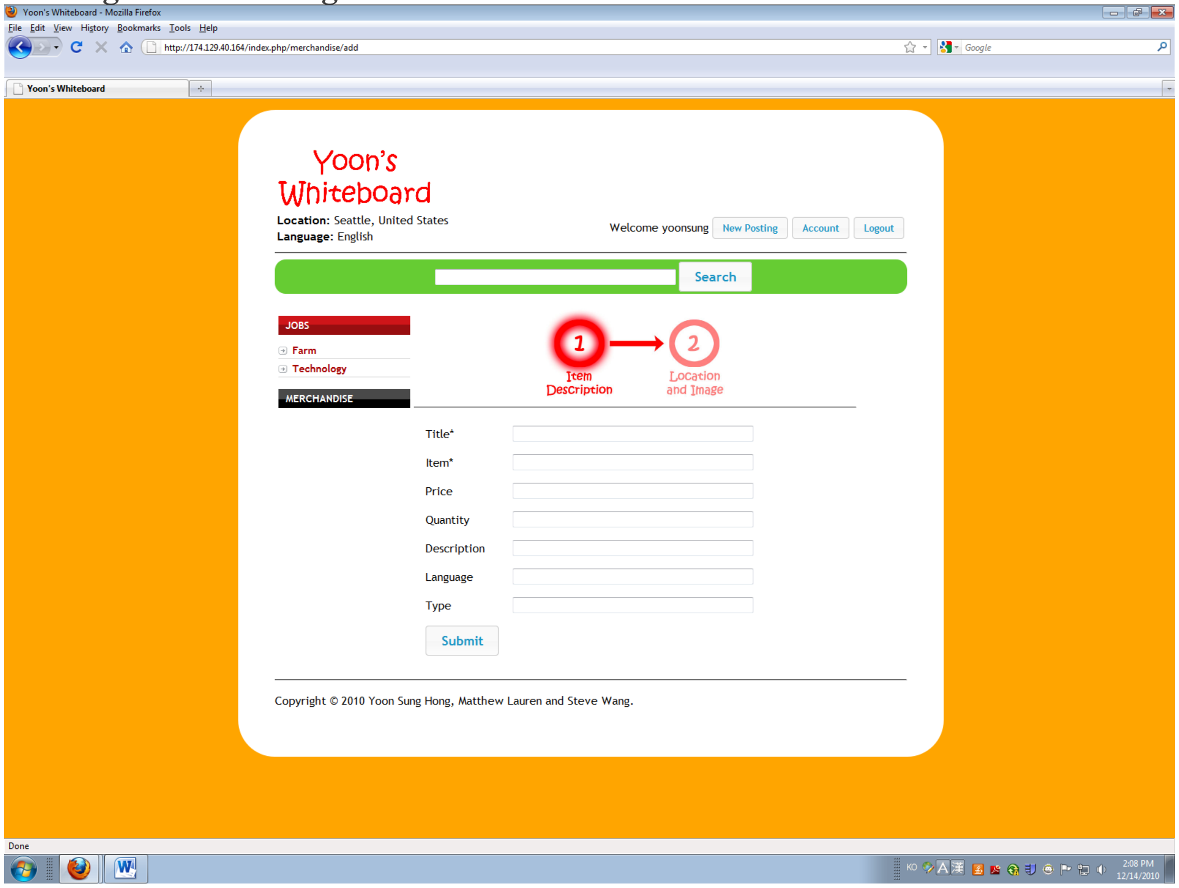
The item page is designed to give the item description and provide users with a space to hold conversations about a specific item (see Figure 7).

The Merchandise item page has three sections:

- Item Details – price, quantity and description about the item
- Location – location of the seller of the item
- Comments – space for interaction between the seller and potential buyers

Continues on Next Page Regarding Website Features

5. Making a New Posting


Yoon's Whiteboard - Mozilla Firefox

http://174.129.40.164/index.php/merchandise/add

Yoon's Whiteboard

Location: Seattle, United States
Language: English

Welcome yoonsung

Title*	
Item*	
Price	
Quantity	
Description	
Language	
Type	

Copyright © 2010 Yoon Sung Hong, Matthew Lauren and Steve Wang.

Figure 8: Process of Making a New Posting

After a user study, we received feedback on our "Making a New Posting" page. The majority of the complaints were about giving users too many input boxes to fill out on a page. To resolve this problem, we split the new posting page in to two and placed a status box to help users track and be aware of where they are in the process (see Figure 8). On the first page, as shown in Figure 8, the user is asked to provide an "Item Description" for the product they are posting. Example input fields are the "Title" of the product, "Price" per unit of the product, and the "Quantity" available. The second page requests the user to provide the location of the product and, if available, an image of the product. The location is simply a postal address. Once the user has completed the second page, we look up and tag the product with its latitude and longitude position and store all of the data into the product database. The lookup of latitude and longitude is performed with the aid of the Google Maps API, specifically its geocoding functionality, which takes an input postal address and outputs the latitude and longitude of the address.

6. Mobile Page

The mobile page provides a way to access our website from a mobile device. Currently, we support a "My Posting" interface where users can keep track of their items along with their associated comments. We believe that mobility can be very useful in developing regions. For the future, we will add more useful features to the mobile version of the website.

Use Case

We believe that showing how a farmer in a developing region can utilize our service to sell a product will allow us to show the full spectrum of our project.

Farmer

Farmers are required to have a cell phone that has SMS capability. Using a structured SMS format, farmers can inform middlemen about the product information such as price and quantity.

Farmer -> Middlemen

Farmers can inform the middlemen about the product information using the following structured SMS message. This SMS schema is used by the ChildCount+ project from Columbia University that is actually deployed in Africa to aid healthcare information operations.

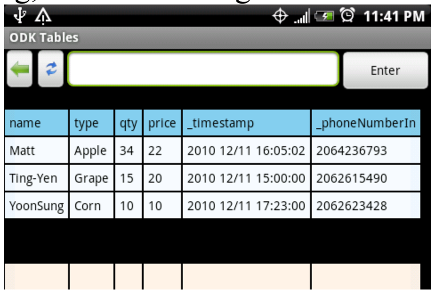
The following is an example SMS message for selling 10 units of corn for $10 per unit by YoonSung:
@Sales +type Corn +name YoonSung +qty 10 +price 10

Middlemen

Middlemen are required to have an Android mobile phone with the ODK Tables and ODK Collect applications installed.

Farmer -> Middlemen

ODK Tables takes all the structured SMS messages with selling requests and saves them in the application's database. Figure 9 demonstrates selling request from farmers Matt, Ting-Yen, and YoonSung, which are being stored and viewed by ODK Tables.



name	type	qty	price	_timestamp	_phoneNumberIn
Matt	Apple	34	22	2010 12/11 16:05:02	2064236793
Ting-Yen	Grape	15	20	2010 12/11 15:00:00	2062615490
YoonSung	Corn	10	10	2010 12/11 17:23:00	2062623428

Figure 9: Example of farmers providing product requests to a middleman

Middlemen -> Server(ODK Aggregate)

The middlemen place postings on the website for the farmers using ODK Collect (example interface in Figure 10). Middlemen have a choice to either make separate postings for each farmer or collective postings. For instance, the middleman can be a village leader who is responsible for collecting products from each household and sell them together. In many farming villages in developing countries, village leaders often carry the responsibility of selling an entire village's farm produce.

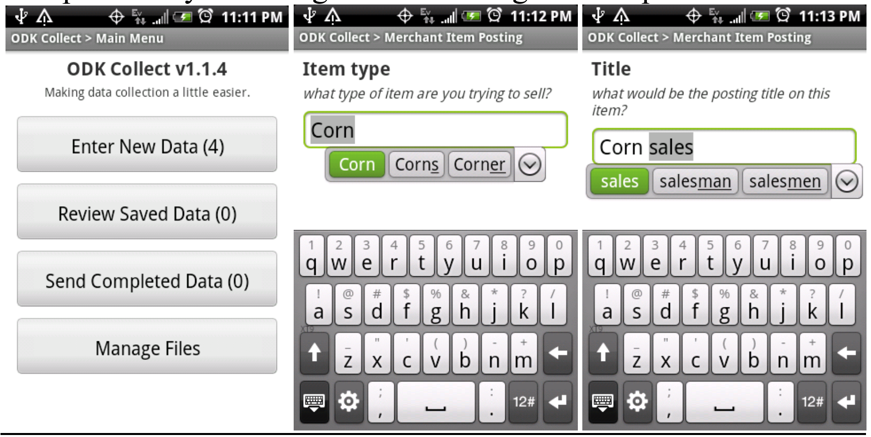


Figure 10: Example of ODK Collect setting up a posting

Server (ODK Aggregate)

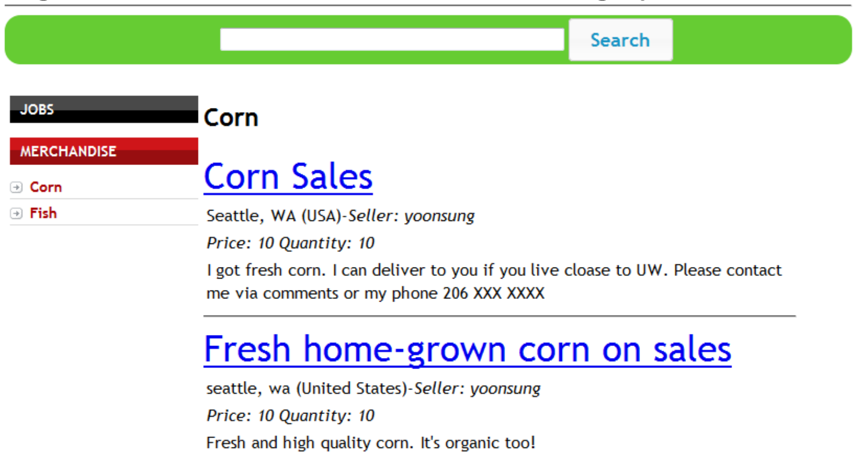
ODK Aggregate, sitting on the Google App Engine, forwards the data sent by ODK Collect to a configured IP address using HTTP POST.

Yoon's Whiteboard

The forwarded data in JSON format is parsed and registered in the database. The item posting is immediately available through our website.

Consumers

Consumers can find items that have been posted either through using the search tool or by browsing through the Categories listed on the left side of the page. Figure 11 provides an example of viewing Corn items under the Merchandise category.



Corn

Corn Sales

Seattle, WA (USA)-*Seller: yoonsung*

Price: 10 Quantity: 10

I got fresh corn. I can deliver to you if you live closae to UW. Please contact me via comments or my phone 206 XXX XXXX

Fresh home-grown corn on sales

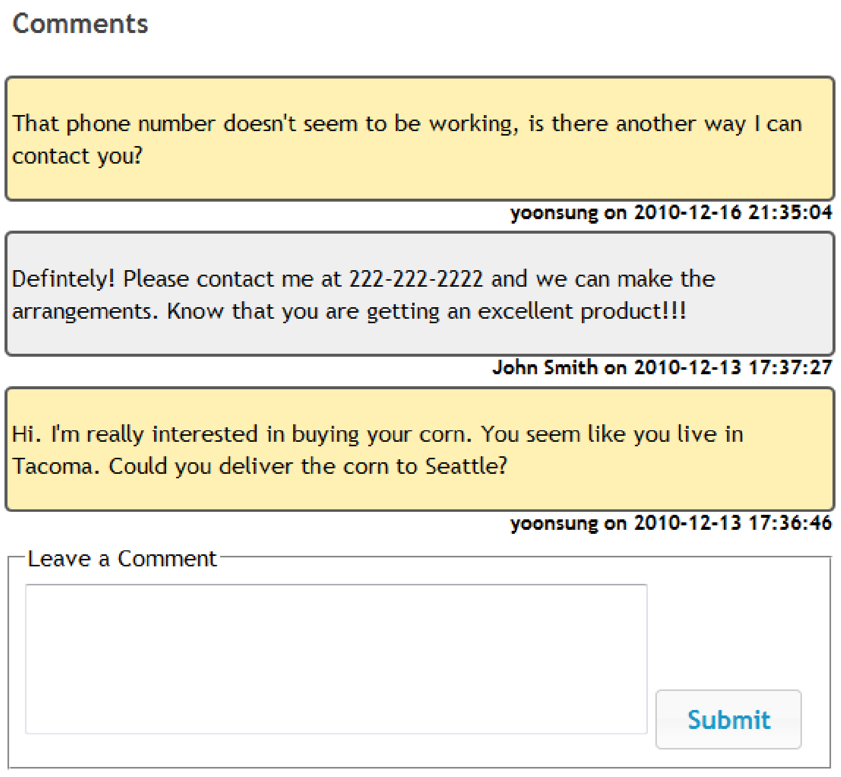
seattle, wa (United States)-*Seller: yoonsung*

Price: 10 Quantity: 10

Fresh and high quality corn. It's organic too!

Figure 11: Example list of items displayed after clicking on the "Corn" category

After the customer has logged onto the site, they will be able to see a box that allows them to post a comment about an item. Their own comments will show up shaded yellow. Figure 12 demonstrates how the logged in user, *yoonsung*, sees all of his comments shaded yellow and all of the other comments shaded vanilla.



Comments

That phone number doesn't seem to be working, is there another way I can contact you?

yoonsung on 2010-12-16 21:35:04

Defintely! Please contact me at 222-222-2222 and we can make the arrangements. Know that you are getting an excellent product!!!

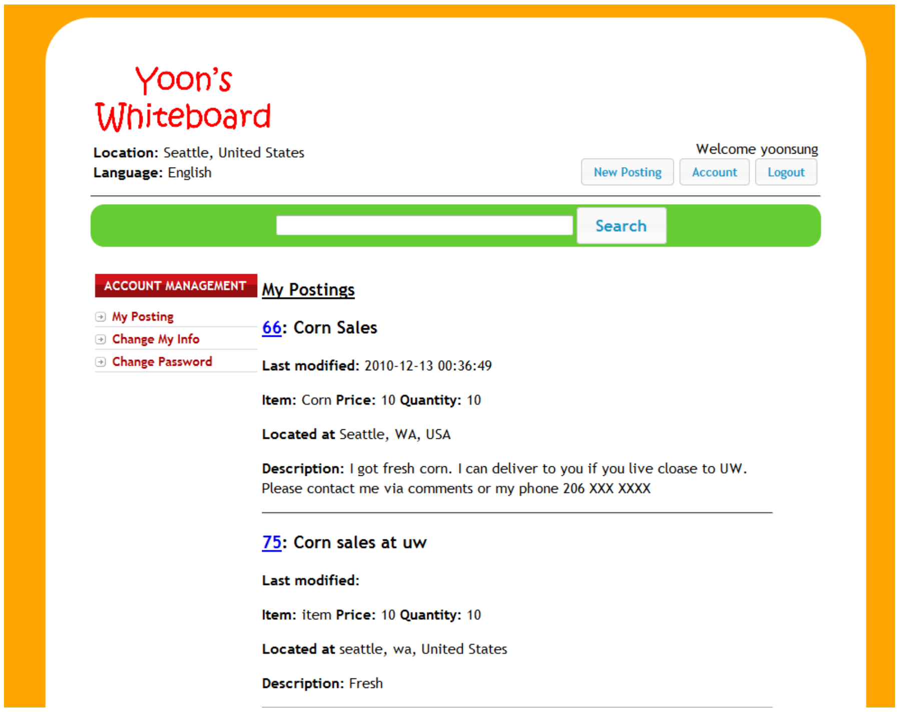
John Smith on 2010-12-13 17:37:27

Hi. I'm really interested in buying your corn. You seem like you live in Tacoma. Could you deliver the corn to Seattle?

yoonsung on 2010-12-13 17:36:46

Figure 12: Example comments for an item

The middlemen can keep track of their postings with the My Postings page on the account management part of the website. In Figure 13, the logged in user (or middleman), *yoonsung*, has navigated to the My Postings link and is viewing all of the postings that he has made.


Yoon's Whiteboard

Location: Seattle, United States
Language: English

Welcome yoonsung

My Postings

66: Corn Sales

Last modified: 2010-12-13 00:36:49

Item: Corn **Price:** 10 **Quantity:** 10

Located at Seattle, WA, USA

Description: I got fresh corn. I can deliver to you if you live closae to UW. Please contact me via comments or my phone 206 XXX XXXX

75: Corn sales at uw

Last modified:

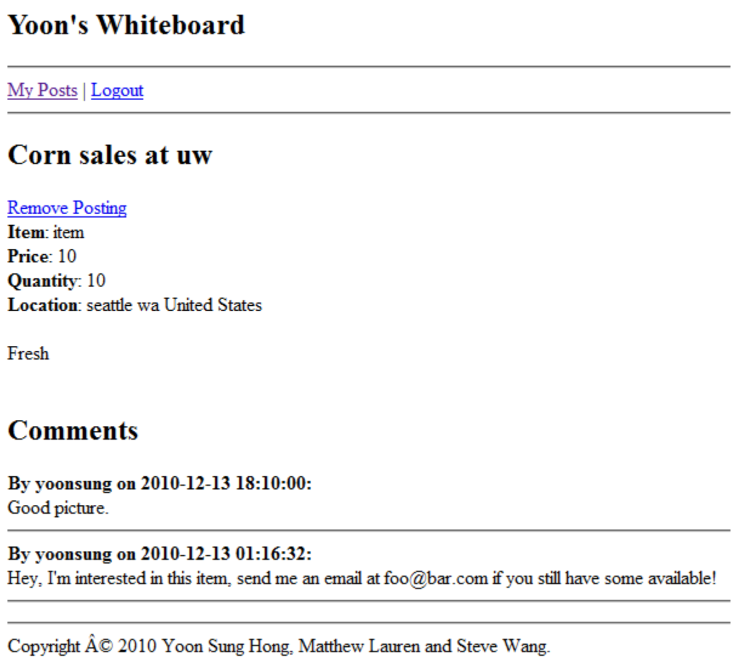
Item: item **Price:** 10 **Quantity:** 10

Located at seattle, wa, United States

Description: Fresh

Figure 13: Example of viewing one's own postings

Also, the middlemen can use the mobile website to view their items from their Android phone. Figure 14 provides an example of what an Android phone's screen would actually show when it tries to view the details of a particular item – in this case, "Corn sales at uw." You can also see in the figure that at this point, the middleman can see that someone is interested in his or her item and they are able to get in contact with the potential customer.


Yoon's Whiteboard

My Posts | Logout

Corn sales at uw

Remove Posting
Item: item
Price: 10
Quantity: 10
Location: seattle wa United States

Fresh

Comments

By yoonsung on 2010-12-13 18:10:00:
Good picture.

By yoonsung on 2010-12-13 01:16:32:
Hey, I'm interested in this item, send me an email at foo@bar.com if you still have some available!

Copyright Â© 2010 Yoon Sung Hong, Matthew Lauren and Steve Wang.

Figure 14: Example of viewing an item through the mobile portion of the website

Experiments

Performance Testing

In order to evaluate our ability to go live with our service, we performed a simple performance test to see the level of traffic our current setup could feasibly handle. Recall that our website is being housed on Amazon's AWS in a small EC2 instance. The approach that we took to evaluate our performance was to utilize Amazon's Elastic MapReduce, which is built on Apache's Hadoop. The mapper function represented a simulated user written in Python and timed how long it took for the simulated user to fetch the home page, perform a search, and fetch the search results. The reducer function called Hadoop's *aggregate* library and performed a count of each unique response time. We measured our website's traffic performance by setting up different numbers of concurrent simulated users, or mappers, and analyzing the resulting list of response times.

At first, in order to reduce our usage of our Amazon AWS credits, we started mapping to small EC2 instances – again, a small EC2 instance is defined as having 1 virtual core with 1 EC2 Compute Unit (equivalent to 1GHz), 1.7GB of memory, 160GB of instance storage, a 32-bit platform, and moderate I/O performance. Due to Amazon rules, we were allowed up to 19 instances – any more required a special request to Amazon. On these small EC2 instances, we simulated 1, 3, 5, 10, 15, 25, 50, 75, 100, 150, and 200 users. From Figure 15, we can see that we can acceptably handle around 10 users at a time, after which the response time is beyond acceptable parameters. This actually makes sense since our website is housed on a small EC2 instance, which means we not only have limited resources and limited I/O, but we are also sharing those limited resources in the cloud. When we decide to make our services available to the public, we will have to provision more server instances and, perhaps, load balance those instances.

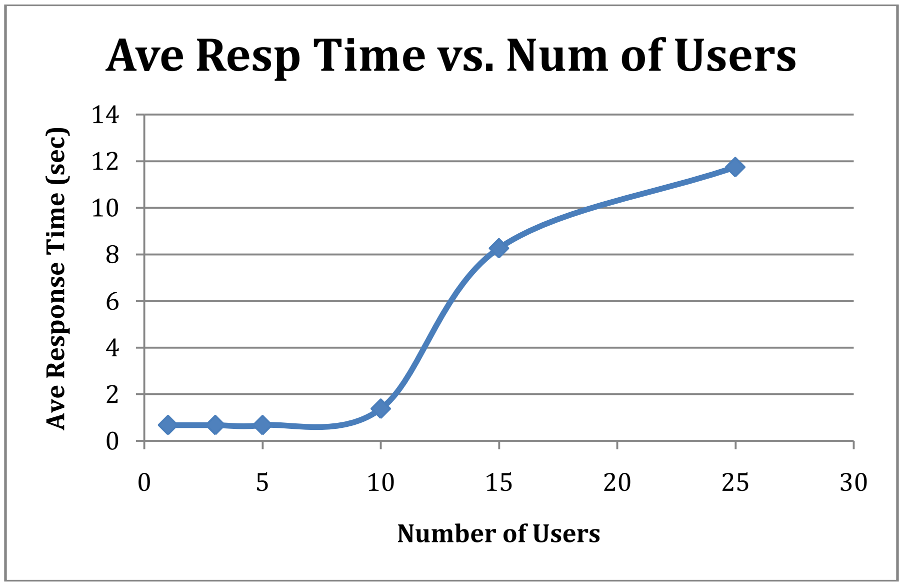


Figure 15: Graph representing average response time in seconds with respect to the number of users from 1 to 25 users issued to Elastic MapReduce with Small EC2 instances

In Figure 16, we can see that the problem gets worse and worse as we add more users into the test. The apparent leveling out of the graph after 100 users may be a saturation point in AWS, such that Amazon is probably provisioning more resources after this point. The data point at 150 users is also interesting in that we get a sudden improvement in performance; this was due to the fact that we ran that particular test at 3AM in the morning when resources in the cloud more freely available.

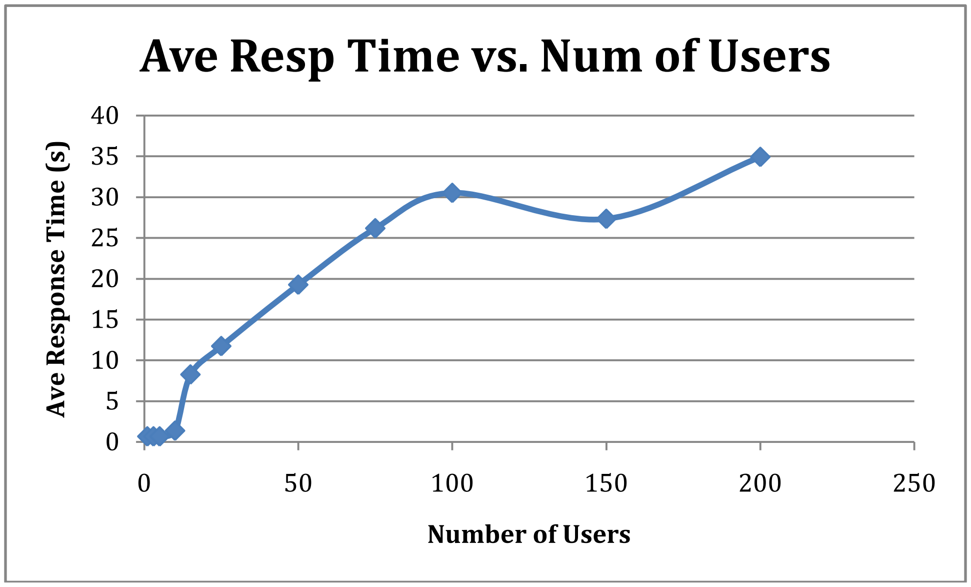


Figure 16: Graph representing average response time in seconds with respect to the number of users from 1 to 200 users issued to Hadoop with Small EC2 instances

After studying the results some more, however, we started to wonder what should happen beyond 19 users and beyond the 19 cores that were available to us. Could it be that beyond 19 users, some of the simulated users were being blocked from starting, providing artificially faster results? In other words, with only 19 cores, if we simulated 100 users, will all 100 users start at around the same time and accurately simulate web server resource contention or will the simulated users block each other such that we are in essence just running about 19 users at a time in serial? If this is true, then Figures 2 and 3 are actually optimistic and our performance is even worse than depicted. In order to analyze the validity of our concerns, we changed from provisioning small EC2 instances to High-CPU Extra Large instances for our Elastic MapReduce sessions. Each of these instances contained 8 virtual cores at 2.5 EC2 Compute Units each, 7GB of memory, 1690GB of storage, a 64-bit platform, and high I/O performance. Of significance to us was the upgrade from 1 core to 8 cores per instance. Due to an apparent problem with AWS, we were only allowed up to 15 extra large instances, but that still provided us with up to 120 available cores with which we tested 1, 3, 5, 10, 15, 25, 50, 75, 100, 120, and 150 users. From Figure 17, we can see that our concerns were validated – our response times were even worse when using the extra large instances and all of the simulated users are allowed to run at around the same time. Note that the 150 users data point may have dipped down because of that fact that 150 users is beyond the scope of our 120 cores and we may be blocking users again. Also notice that the extra large instance exceed the small instances in response time only after around 50 users; this may be because of the greater capabilities of the extra large instances (they are over twice as fast as the small instances with greater I/O performance) that are eventually overcome by the number of concurrent users and the immensity of requests being handled by the website.

In conclusion, these results demonstrate to us the need to provision additional server capacity before we can offer our website to the public. No matter what type of EC2 instance we used, our Hadoop study clearly shows us that we are currently unable to service any sufficient number of users at the same time.

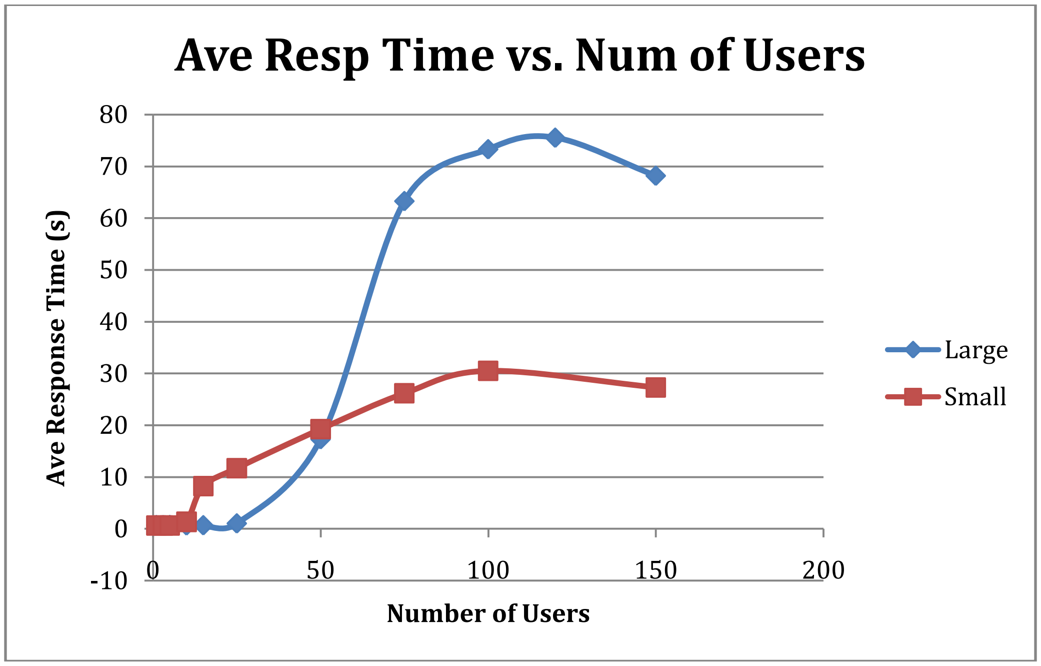


Figure 17: Graph representing average response time in seconds with respect to the number of users from 1 to 150 users issued to Hadoop with both Extra Large High-CPU EC2 instances ("Large") and Small EC2 instances ("Small")

User Study

The goal of the user study was to test how easily a person can come to the website portion of the project and, with no instruction, use all of the features. We decided to limit the scope of the study to the web portion because of limited time and resources and because the mobile portion requires the user to install an app, register, and possibly get some training in its use. The website on the other hand is made to be used by anyone interested in interacting with suppliers, so it must be intuitive and easy to use, which is what this study is focused on.

Test

The study consisted of giving seven people of varying technical backgrounds five tasks to complete. The data gathered was an observation on how the user completed the task compared to how we designed the tasks to be completed. We encouraged the user's to put the same amount of effort in completing the task that they would put in using any other commercial site, such as give up when it just is not worth it.

The specific tasks and our expected paths to completion are as follows:

- Sign Up and then Sign In
  - Click on Sign Up button
  - Fill out required fields
  - Click Submit – See Signup Successful Page
  - Click Login Button
  - Input Information – Click Submit
  - See the "Welcome username" at top of screen
- Find the highest cost corn item by browsing for it
  - Select Merchandise Sidebar Item
  - Click on the Corn item
  - Scroll through listed items, scanning the price field.
- View details about an item with the phrase "img" in it, where a single matching item exists.
  - Type in "img" in the search bar
  - Click on result in search results
  - View Item
- Comment on an item
  - Scroll to bottom of page
  - Type in what you want to say in the "Leave a Comment" box
  - Click Submit
  - Scroll down to comments section again, view your comment.
- Post an Item
  - Click on "New Posting" button at top of page.
  - Fill in required fields.
  - Click Submit.
  - Fill in more fields.
  - Click Submit again.
  - View Item

Results

The results of this user study provided very good insight into what we had been overlooking, and what we needed to focus on. In addition to recording task completion, we also recorded any comments the user made while performing the task. The following results table shows the completion progress of each user that attempted a scenario with each row indicating a specific user (see Figure 18). Green means the person completed the task in an acceptable timeframe and in an expected way. Yellow means the user completed the task, but either not quickly, or not in the expected manner. Red means the user was unable to complete the task. The "Fine" entry signifies that the user completed the task as expected with no significant comments. As Figure 19 evidences, our original implementation suffered from a number of human interaction issues, as opposed to bugs, as the majority of the comments dealt with understanding or working with our interfaces: for signing up for an account, one user had trouble finding the button to perform the action; for browsing through a category, a couple of users found the text to be either confusing or unattractive; for searching for an item, a couple of users had trouble interpreting the proper usage of the tool; for commenting on an item, a couple of users could not intuitively interpret how the comments were organized; for posting a new item, a number of users did not seem to understand what all of the fields meant. While Figure 18 provides very useful qualitative results of the user study, Figure 19 attempts to provide some more quantitative results. The bar graphs in Figure 19 provide us with the additional information that our original implementation performed pretty well with the simpler user tasks of the website, like signing up for a user account and searching for an item. However, the harder user tasks, like posting a new item, desperately needed to be looked at and made easier for users.

User Study Results

Sign In	**Browse**	**Search**	**Comment**	**Post**	**Other Comments**
Fine	Fine	Fine	Comment at top not directly intuitive	Type to Category not intuitive, language, type, id and name not showing, listing disappeared, very confusing	
Tried first in IE, didn't work as expected.	Noticed that the Farm/Tech pages are not complete.	Mistyped phrase, no results returned was confusing.	Got errors when not logged in.	Gave up after first submit.	
Fine	Merchandise Listing very confusing.	Fine	Fine	Too unsure about what to put in all the fields on the first page.	"Looks better with javascript disabled"
Fine	List was too fat and wordy to browse.	Fine	Fine	Can't find category field. Wondering why it's not all on a single page.	
Looks odd in Chrome.	Fine	Looked around for an advanced search option, to search only a specific part of the item	Looked at top of comments list to post one, then scrolled down.	Fine	
Fine	Fine	Fine	Fine	Fine	"Intuitive but too ugly to use".
Looked awhile for Sign Up button.	Took longer than expected, eventually made it.	Gave up at search page	Already given up	Already given up	"Shouldn't have asked me, I'm not good with computers."

Figure 18: Qualitative results of the user study

User Success / Difficulty / Failure Ratios

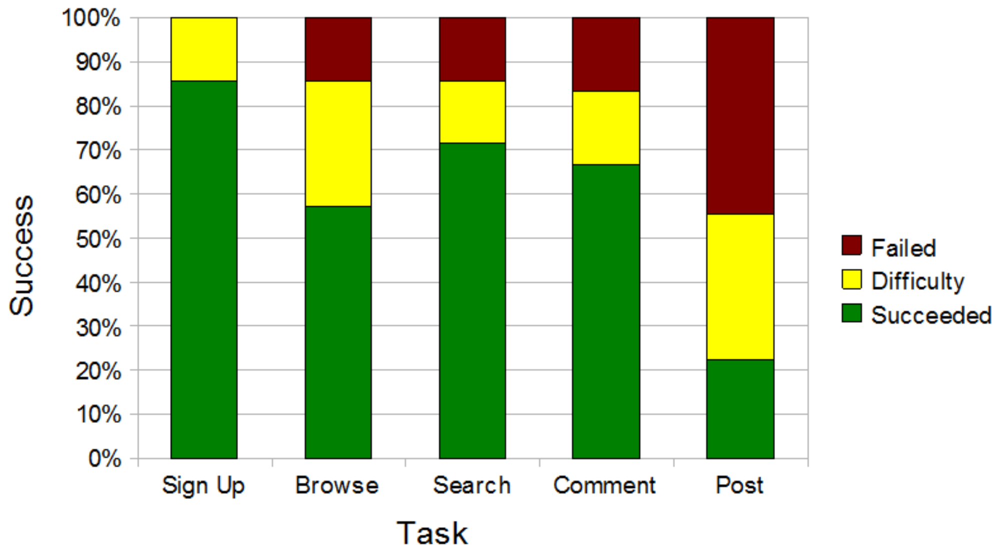


Figure 19: Quantitative results of the user study

Improvements

From these testing results, it was clear to us that we had been making too many assumptions about how people use web pages and that we needed to devote a lot more time than we had left into look, layout and workflow. Specifically, we got a lot of feedback on the look of the site – that it was just too plain and needed to be more eye-catching. This was one area that we tried to remedy. We did a pass over the look and layout of the site, improving proportions, adding colors, and making the different pieces of the site stand out from each other in order to give the eye quick things to latch onto. Other improvements we implemented are to highlight one's own comments, making one's comments more visible, and making the steps for posting an item more apparent and logical.

Going forward, we will be continuing to make improvements to the site keeping this feedback in mind. One specific change that we did not have time to do, would be give all input fields descriptions, which would reduce confusion and give the users confidence in what they are doing. We would also do further, more in depth, iterative user tests to determine further areas of the website that needs work. Finally, we would eventually like to provide a brief tutorial on how to do the common tasks on our website, since the users we tested told us it would be very helpful.

Surprises and Lessons Learned

Surprises

Throughout the course of the project, there were a number of surprises – things that ended up being harder than we had anticipated. Perhaps the biggest surprise was browser compliance. We did most of our work with Mozilla Firefox and used that browser as our baseline. However, we discovered more and more that adapting the site to work on other browsers like Chrome or Internet Explorer was a major challenge sometimes seemed like a project unto itself. Developing our search system to integrate location into the standard text search was another challenge that ended up taking most of the quarter to solve. We always planned to make location the central aspect of search, but we could not figure out to properly integrate full-text searching; more specifically, the problem that we were facing was determining how to properly score text matches and then how to adjust that score based on location. Indeed, for much of the quarter, we were taking the results from a basic prefix search and simply ordering the search results by distance from the user. Professor Weld advised us that databases like Lucene sometimes came with a text search scoring algorithms, which gave us hope for a solution. We continued to look into MySQL for something similar and found the FULLTEXT index that helped to solve our problem.

As opposed to implementation issues, a tool that surprised us was using Hadoop with Elastic MapReduce. Just to get it set up was a long process of not just signing up for the Elastic MapReduce service, but also signing up for Amazon S3, which MapReduce uses to store mapper functions, input files, and output files. Then, since only the owner of the AWS account could use the MapReduce and S3 GUI consoles, we had to set up the MapReduce command-line interface (CLI) and an S3 command line tool so that everyone in the group could use Elastic MapReduce. The MapReduce CLI did not work at first, as it required a specific version of Ruby that we were not using – we had installed the latest version of Ruby, which had deprecated syntax that the MapReduce CLI was utilizing. Finally, we could get started with setting up our Elastic MapReduce session, which involved learning the mapper syntax, the *aggregate* library, and the MapReduce and S3 command-line syntax. We also had to learn how to translate the sometimes cryptic status messages and log files since we ran into a number of errors before we were able to get some sessions started – the most difficult of the errors being the undocumented limit of 15 high-CPU extra large instances, even though the documentation states that we could provision up to 19 instances as was the case with the small instances. A final surprise that we would have never expected was the difficulty in implementing image input for product postings, which can be done with both the web site posting process and with the smart phone. Basically, we had a lot of difficulty figuring out how to first correctly encode and store the images and then decode and display the stored images.

Lessons Learned

In doing the capstone project for CSE454, we received a number of lessons learned – some of them being lessons of things that worked well and should be carried forward and other being lessons that did not work well and need to be addressed. One of the most significant things that helped us in the project was incrementally setting milestones and meeting those milestones as a team. This kept us moving along step by step, both preventing us from having to experience any major last minute hurdles and allowing us to accomplish most of the goals we set out to accomplish at the beginning of the project. One of those milestones was arguably one the most beneficial actions that we took during the course of the project: identifying the design, programming languages, and technologies early in the project. These decisions did not come easily nor quickly and we have to spend many hours in team meeting sorting through all of the options and debating the benefits and drawbacks of each option. By the end of this milestone, we had identified LAMP are our architecture (therefore, PHP would be our language), chosen CodeIgniter as our coding framework (which uses the MVC framework), decided upon AWS as our platform, and had initial designs of our database schema and code. Putting the time and effort into these initial decisions saved us countless hours by preventing major design changes midway through the project. Choosing CodeIgniter and the MVC framework was also something that worked really well for us. The framework helped to neatly divide up the major pieces of the system. This allowed us to more easily visualize how to organize our code, but it also helped us to work well concurrently since the framework helped us to know exactly how everyone's pieces interacted and fit in the system. Using AWS also turned out to a good choice for the project. AWS turned out to much easier to set up than we anticipated, provided virtually 100% uptime from what we could tell, and placed a number of handy tools at our fingertips, such as MapReduce (though it was hard to set up, it was nice to have a complete Hadoop setup on hand). Finally, like our use of CodeIgniter and AWS, a good lesson that we learned from doing this project was that using the right tools can tremendously aid our development. For example, we could not have come close to the level of success that we achieved with location identification without the help of tools like IPInfoDB and Google API's geocoding.

As opposed to many lessons concerning things that worked really well for us, there were also lessons concerning things that did not work so well and need to be adjusted or fixed. One thing that was definitely a challenge and something to keep in mind as we move forward was the lack of web user interface (UI) experience in our group. We originally had a UI focal at the beginning of the project, but the focal had to drop out of the class a couple of weeks into the quarter. This made UI tasks fairly time consuming as most of us were learning how to do UI while we were developing. If we could do the project again, we would have looked for someone with some UI experience to join the group and will definitely be an item to address as we continue to develop the website. Another painful lesson to take away from the project was the lack of test planning at the beginning of the project, which led to the lack of any form of a test harness. Throughout the project, we just naively performed manual testing whenever we made a change; however, as the project got more and more complex, this manual testing not only took more and more time, but was only prone to missing more and more errors. Similar to the lack of a test harness, we also mistakenly left out the creation of test environments at the beginning of the project. This caused us to often step on each other's code as we developed straight on the production code. Most of us eventually developed test environments, but this is something that we will have to remember to keep on addressing. Finally, although obvious, late changes never worked well. Although we put in a lot of time and effort into foundational design and decisions at the beginning of the project, which really did help to avoid major changes, we still experienced a couple of small changes that still cost of precious time. For example, we experienced a deprecation project in the middle of the project that took about two four full days to complete. The lesson is that we need to not only continue to devote adequate time at the beginning of projects for design and making key decision, but we need to keep getting better at making these decisions so as to avoid late changes more and more.

Conclusion

Our group is very satisfied with the results of the project. We managed to achieve all the goals we proposed at the beginning of the course. We believe that we have completed the groundwork required to extend the project further. We ask advice from academia and experienced software engineers to design and implement a sustainable system. For the immediate future, we would like to perform user studies on local farmers in Seattle to see if the farmers find incentives to use the system. Although the system is designed for developing regions, we believe that our system can also be useful in the developed countries context. Once the web service stabilizes, we will like to expand our service to developing regions and perform full stack investigations from mobile to web. It is our plan to work closely with academia with interests in this context to get advice and find the first appropriate region to deploy the service.

Appendix

Division of Labor

Yoon-Sung Hong carried the vision of the project and led the team, especially in terms of the design and architecture of the system. He was also in charge of the mobile portions of the system, which included developing the cell phone to smart phone interface, the smart phone application, and the website's mobile interfaces. Yoon-Sung was also in charge of the admin system of the web site, which included the login system

Matthew Lauren was in charge of user interface (UI) design and development, which included designing the overall CSS and page layout. He was also in charge of the customer interaction portions of the web site, which included the product viewing and commenting systems. Lastly, Matthew also ran the user studies.

Ting-Yen Wang was in charge of developing the database schema and maintaining the database. He was in charge of designing the website's search system, which included working on the IP to location library, the geocoding tools, and the addition of location into natural language full-text search. Lastly, Ting-Yen was in charge of the performance testing.

Overall, the group dynamics were very positive. We met about twice a week to discuss design decisions, past week accomplishments, and tasks for the upcoming week. As expected, we would sometimes have differences of opinion, but these were always a pleasant times of teamwork to get at the best solution and never battles over individual stances. Outside of meetings, we communicated on a consistent basis by email, instant messaging, or phone. One of the major aspects of our group was that we regularly set milestones and kept each other accountable to those milestones. By the end of the project, we all felt that it was a good group experience that we would gladly repeat.

External Code

- **CodeIgniter:** PHP framework that both helped us to use the Model-View-Controller framework and provided useful libraries and function for web development.
- **ODK :** A series of Android applications that help to collect SMS messages, display the messages as a form that can be further manipulated, and then submit the form to a web server.
- **IPInfoDB:** Online database that provide mappings from IP addresses to location information, such as latitude, longitude, and region codes (such as city, state, and country)
- **JQuery:** Provided a set of JavaScript functionality that helped to simplify some of the more complex look-and-feel items of the website. For example, the spinner that appears after posting an item was done with JQuery and the confirmation box that appears after clicking on the "Remove Item" button was done with JQuery.
- **JQueryUI:** Provided some UI plug-ins to help with UI design. We largely utilized the buttons library to improve the look-and-feel of our buttons.
- **Anthing Slider:** JQuery plug-in that helped to implement the "slider of suggested products."
- **According Content:** JQuery plug-in that helped to implement the left-hand-side menu bar.
- **Google Code API:** The Google Map, map markers, and geocoding was done with Google APIs.
- **AWS Elastic MapReduce:** Amazon's Elastic MapReduce is built on Apache's Hadoop, which we utilized to do our performance tests.
- **S3cmd:** Command-line tool that helped to interface to S3 without the Amazon GUI management console.
- **Elastic MapReduce CLI:** Amazon's Ruby MapReduce command line interface that helped to interface with MapReduce without the Amazon GUI management console.

How To Use Our Code

Environment Requirement
Browser: Firefox
OS: Linux / Mac OS
Server: Apache Http Server
Language: PHP

Sample Web Site: ***http://174.129.40.164/index.php***
Mobile Portion of Web Site: ***http://174.129.40.164/index.php/mobile***

We assume that you have set up Apache server environment locally with a working version of PHP and MySql with the server.

1. Unzip file
2. Place the folder(yoonswhiteboard) at the root of Apache server directory
3. Set up the database by executing the SQL scripts in the file DBShemaDescription.rtf
4. Configure the database in the system/application/config/database.php file
   4.1 Replace hostname with 127.0.0.1 if MySql server running locally
   4.2 Replace the database username with yours
   4.3 Replace the database password with yours
5. Add an "uploads" directory at the root of the web server if not existing (at the same level as the "system" directory)
6. Use Firefox browser and go to 127.0.0.1/yoonswhiteboard
7. Now you should see the first page of our web pages.