Freshipes

Min Sul, Panharith Thong, Nicholas Brekhus
CSE454 – Autumn 2010

Project goals

Our primary objective in creating Freshipes was to make it easy to buy ingredients for recipes one finds online. To facilitate this goal, we curated some recipes at a site of our own design, this gave us the opportunity to improve the user experience and learn about building websites. It also allowed us to restrict the set of recipes to ones we had good metadata for.

System Design

Infrastructure

Our web application was developed atop mysql and php on apache linux (LAMP); we had originally intended to use django on apache, but our lack of familiarity with the environment caused us to abandon it before we started breaking ground. The front end code heavily used jQuery and jQuery plugins/snippets.

We had several offline processing jobs, for which we used python as the primary language. These jobs depended on various external libraries (lxml, scrapy, py-wikimarkup, …) and were used for data acquisition and preparation.

Application Design

Web Application

Front End

The front end of the web app was done in PHP and made use of Javascript, JQuery and CSS. The choice of design was to captivate the users that browse for recipes and understand their needs. To captivate, we want to show pictures of recipes because that's what attracts people to browse our site. The users need to find recipes fast, the big search bar on the page and we want to them to discover and try new recipes (new recipes, featured section, top rated and recommended section). For that used JQuery to display the recipes in a dynamic way so users can rollover on a recipe and find information quickly about that recipe and whether they are interested or not.
The autocomplete search feature makes use of JQuery, AJAX and PHP to retrieve data in the database in the background from the user query input. The results from the database are passed back in JSON format to JQuery to display the search suggestions. Typing a search that didn't match any specific recipe will display a result page where it lists all the recipes that contain the query searched. JQuery and CSS are useful to display a

nice formatting of recipe results, where the next batch of results is scrolled up. We get the result by doing a MySQL query from PHP and display the results in an HTML formatted. Each recipe name is linked and uses GET to pass the recipe id to PHP to display the detailed recipe page. Each recipe in the list of recipe results will have a small description and an average rating. Clicking on the recipe link will bring to the detailed recipe page.

In the detailed recipe page, we use PHP to get the recipe id variable passed in the URL, we can query the database for the image, recipe rating, description, ingredients, descriptions, reviews pertaining to that recipe. The ingredients name have attributes "url" and "rel" to help JQuery figure out which ingredient name the user clicked on which then pops up an overlay iframe of Amazon Fresh for the user to buy that ingredient and other ingredients needed for that recipe. The ingredients are retrieved from the "tag" table in the database which contains the entire ingredients name for that recipe. A facebook plug-in was inserted on that page to let users comment and also post on their facebook profile, as a way to advertise the site.

One of the distinct features is the "Recipe Wall" page, where it lists 4 by 5 recipe thumbnails for users to discover new recipes they might like to try. This is again done using JQuery and PHP where it queries MySQL to pick 20 randomly selected recipes and display them on the recipe wall. JQuery makes it so we can click on each thumbnail to see the description and if the user likes it they can go directly to that recipe page for more detailed directions and ingredients to use.

Database Schema
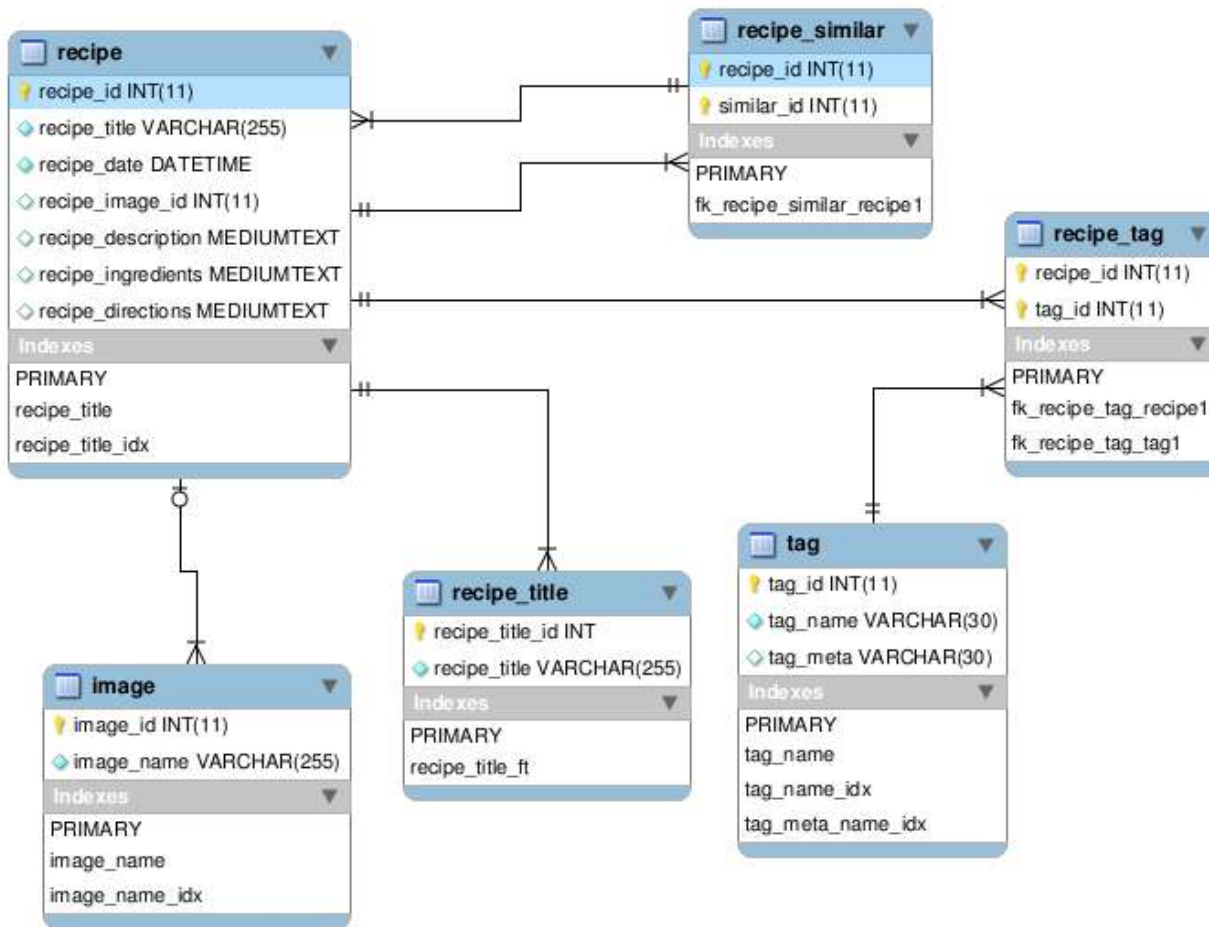
Scraper

Using python's various modules and libraries along with Scrapy, an open source scraper/crawler, we were able to create a scraper to obtain substantial data, or at least as much as we could obtain and connect with our recipes. Regular expression was our main component in scraping for our data and scrapy helped organize and export the data, especially converting "items" gathered through scraping to xml format which can be converted and imported into database.

Scrapy was mainly used to scrape the reviews off of allrecipes.com which contained large volumes of reviews. First we decided to obtain the most popular recipes or the most reviewed recipes since it will help us determine which recipes are recommended by which reviewers and what other recipes these reviews also liked. Allrecipes.com had list of all the recipes and we could order them by their ratings in descending order. With this the spider can be created with a seeded url of the first page of the list of recipes then proceed to scrape the recipes and their reviews page by page. And because of the way the list was structured it was easy to control how much reviews about each recipes to scrape. We decided that about 3-4 pages of each recipe was enough for our purposes. In total we scraped over 150,000 reviews which we deemed was more than enough for developing and testing our recommendation scheme and algorithm namely slope 1.

We had little bit of trouble scraping images since a lot of them were copyrighted and pages in wikia that contained images we wanted were not as well structured as we'd like. Fortunately, Nick found that we could obtain images through using wikia's url scheme. Disappointingly however, when scraping images through scrapy, we found that it hashed the names which we could not use since we needed to connect them with our recipes. So we ended up using simple python modules and libraries to read list of recipe names from the data dump Nick created and simply scrape/download images one by one. We were able to obtain over 2,000 images in the end which made our site considerably more tasty.

Exporter

Nick wrote the exporter, which took a mediawiki db dump from recipes.wikia.com/wiki/Special:Statistics and translated it to our schema. To accomplish this goal he structured the exporter program into two separate parts.

The first part of the program is a 'translator' which leveraged lxml and py-wikimarkup to parse the mw xml dump. The translator also contained a lexicon based IE system which tagged recipes based on the targets of their internal wiki links.

The second part of the program was an exporter, which was designed to be flexible in the event our table schema changed (and it did). It also had a review dump target, to fold the reviews Min scraped into our database.
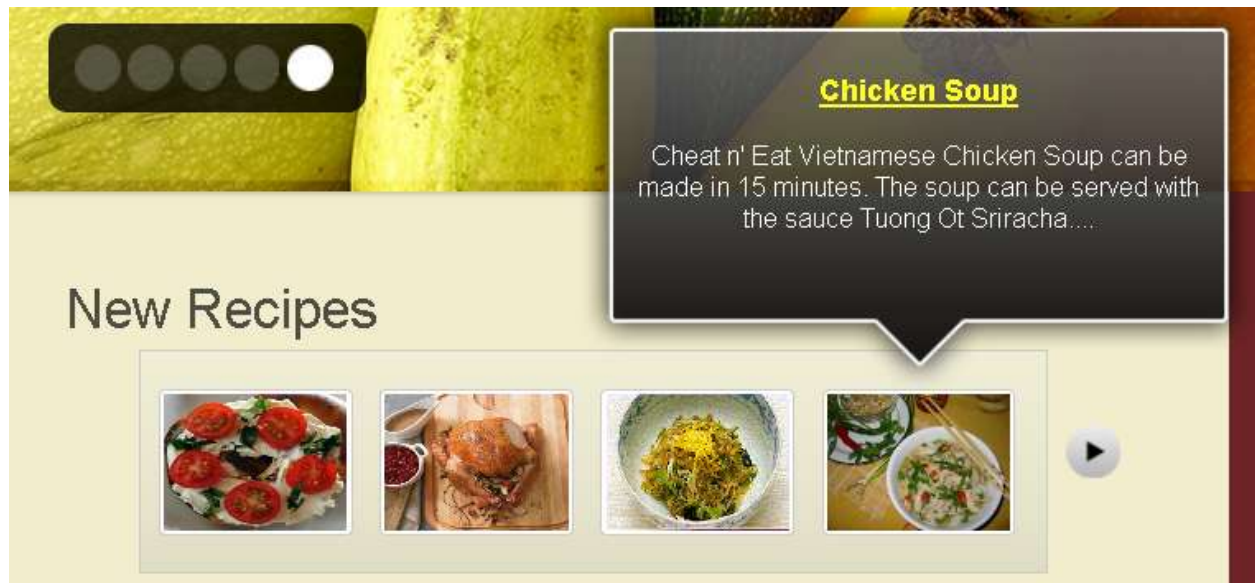
Typical Usage Scenarios:

Above is a screenshot of the home page of our recipe site. There are several components to note. First, it features in the center a search bar where users can search for recipes by title and also by ingredients.
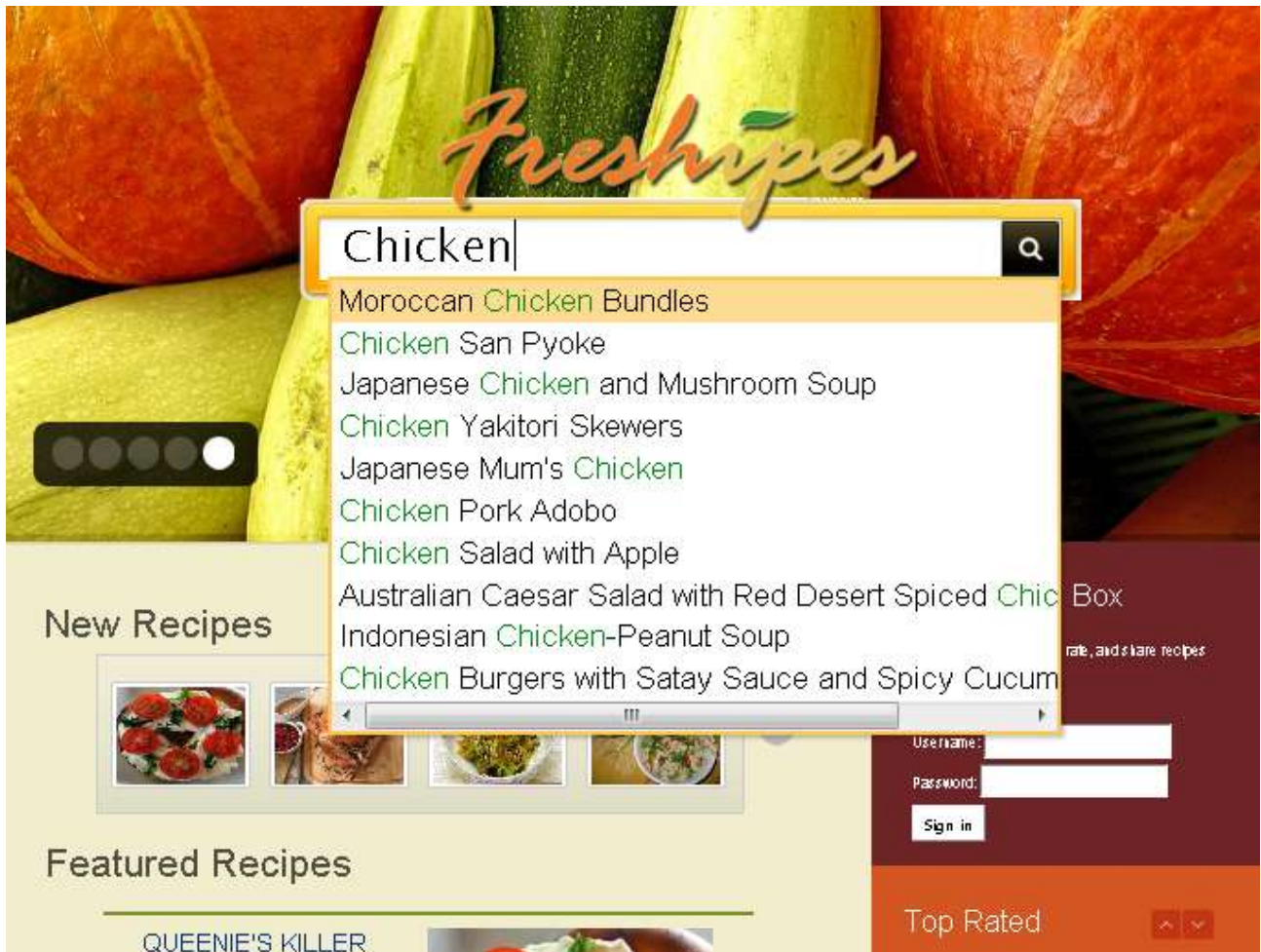
The search bar has autocomplete where it will list all the recipes matching the text input in the search bar when the user types.

A section "New Recipes" where we list the 12 latest added recipes that have a picture. The user can rollover each thumbnail of the new recipe and preview the description. Clicking on the right arrow will display the next batch of 4 newly added recipes.

Below this, we have the "Featured recipes" where the site administrator can pick a few recipes to be featured considering they match the season (i.e. Thanksgiving recipes, Christmas recipes, or other occasions). They are also scrollable left and right to view the previous or next recipes respectively. The "Recipe#1", "Recipe#2", "Recipe#3" tabs below it can be clicked to quickly access a specific featured recipe.

On the right side, we list the 3 top rated recipes based on the user rating.
To view the next top rated recipe, the user can click on the up or down arrow.
The recommended section usage is identical to the top rated section but displays only the recommended recipes.
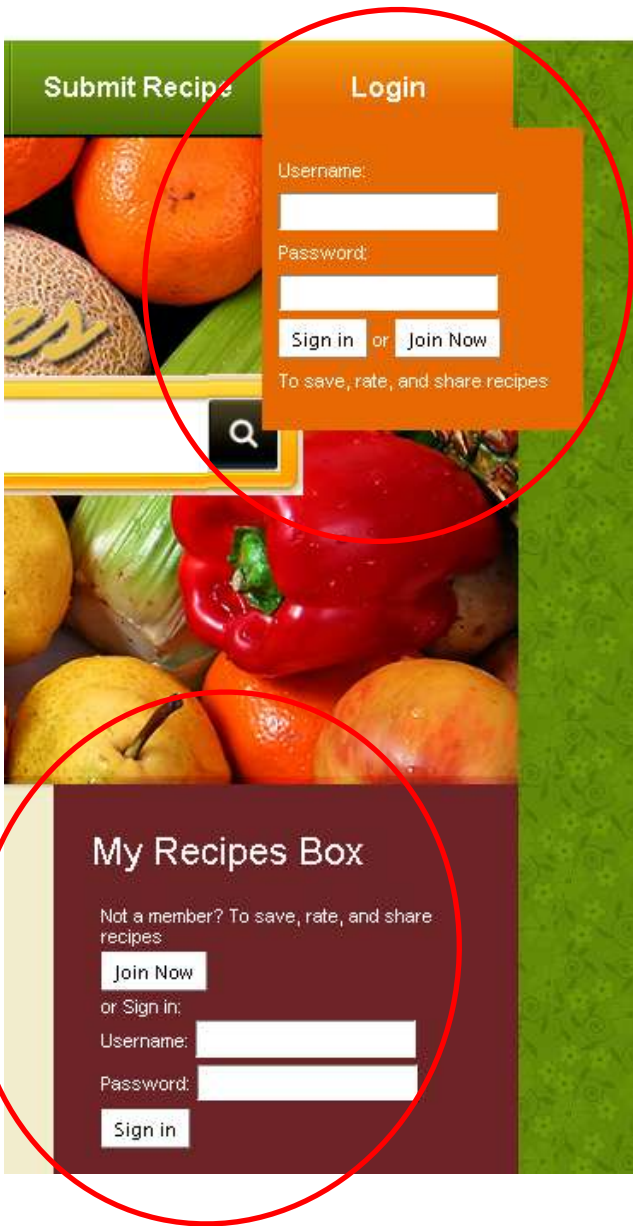
The login form will show up if the user hasn't already signed in. Users can sign up from 2 different places, from the tab and from the right section "My Recipes Box". Or if the user is a new member they can click on "Join Now" that will redirect to a sign up page.

The "My Recipes Box" when logged in will show the saved recipes of that user and show a log out link.

After the user search a query in the search bar and doesn't match any specific recipe in the database, it will show the search results matching the query with the name of the recipe, a description and the rating, where the user will be able to browse for the recipe they're looking for.

When the user finds a specific recipe match, the recipe page will show the detailed recipe with the image, rating, description, all the ingredients needed for that recipe and the directions. The ingredients name have link. When you click on them it will pop up an iframe to purchase that ingredient on Amazon Fresh. It will overlay the current page with a close button.

The iframe has the recipe name and the ingredients that are needed for that recipe. The user can click on the name of ingredients to buy to see a list on amazon fresh with different price where the user can choose to add to cart. After clicking on the ingredient it will remove that ingredient from the list to buy.

Reviews

(use arrow keys up ↑ or down ↓ to see previous or next comments)

« Previous Reviews   Next Reviews »

🥄🥄🥄🥄🥄 by Nicole Pasquini on 2010-11-27
Love this recipe! Only one change - I use 3/4 lb crimini mushrooms and about 1/2 lb shitake - the variation adds more flavor and texture.

🥄🥄🥄🥄🥄 by westcoastcook on 2010-11-25
I halved the recipe (there are only 2 of us) and used different mushrooms. Otherwise followed the recipe and it was excellent. I actually wish I'd made the full recipe so we'd have left overs for the next day. I'll definitely make this again. Also I did heat my broth and had it simmering on low while cooking the other ingrediants and the cooking times did work as per the recipe. Thank you for submitting this!

🥄🥄🥄🥄🥄 by paulam on 2010-11-25
I ended up using 7 cups of broth instead of 6 and omitted the chives at the end, it was amazing. Tasted just like something you'd order in a restaurant in Italy!!! I didn't need to add any salt or pepper either. Only used 1 tbsp of butter instead of 4.

🥄🥄🥄🥄🥄 by Eddie on 2010-11-25
Best risotto I've ever had. Few changes that I make. I eliminate the white mushrooms and instead use 5 large portabello mushroom caps. I sautee the mushrooms a little longer than suggested. Also use real chicken stock. Amazing dish.

🥄🥄🥄🥄🥄 by Tom on 2010-11-21
Excellent! I followed the recipe except I reduced qty of mushrooms to 1.5 lbs total and used mix of portobello, crimini, chanterelle and dried porcini. Also I used the broth from reconstituting the porcini plus Pacific brand Mushroom Broth rather than chicken. Like others, I added some garlic with the shallots. I doubled the recipe for party of 12 -- everyone loved it!

At the bottom of the detailed recipe page, we have the "Review" section where the user can browse for reviews and ratings about that recipe. The user can also post comments for that recipe by logging in facebook.

Create An Account

1. Create Account	2. Contact Information	3. Finalize

Step 1: Username - Password

Please enter your login information:

1. Enter Your Email Address *

Your password will be sent to this address. Your address will not be made public.

2. Pick a username

Your preferred username to be used when logging in.

3. Choose a Password *   Verify Password *

Must be at least 8 characters long.

Step 2

Users can create an account by clicking in "Join Now" link. The registration form has 2 steps before the account is created.

Recipe Wall

This New Mexican inspired recipe makes a great filling for low-carb tortillas and omelets. Depending on your CCLL, you may mix in Monterey jack Cheese, lime juice, sour cream, tomatoes or cilantro to taste.

Calabacitas

The recipe is a useful feature that list many recipe thumbnails for users that are undecided about what to cook. Each thumbnail is clickable. When clicked on, it will show the description of that recipe. If the users want to know more about that recipe, they can click on the recipe link name.

Experiments

We conducted one experiment, a user study, during finals week. We used google docs to create the survey form, which had 3 tasks and asked a total of 19 questions. 12 people responded to our survey. The survey can be found at the following url: [goo.gl/Eu9lj](goo.gl/Eu9lj). Contained within the archive is a pdf named freshipes_user_study_summary.pdf which has the complete summary. The survey was aimed at gathering data which would be useful if we were to further refine the product. The main take aways from the user study were as follows:

- Most people had a good impression of the site when they first arrived at it, after using the site this had somewhat decreased, but was still positive overall.
- The visual appearance of the site received high ratings but many people complained about font sizes, text flow and spacing.
- The task people had most trouble with was adding ingredients to their amazon fresh cart; there were numerous issues and many found the interface unintuitive and confusing.
- Most people did not have trouble navigating within the site but they felt the search feature produced suboptimal results.
- A lot of people had suggestions about how to improve the auto-complete feature and at least one user found it confusing.

The data suggests there is a lot that we could do to make the site better but also shows that the site is at least usable in its current state. We view this as a win. If we were to continue development on the site a more measured approach would be taken in the areas noted above to tighten up the presentation (particularly the results page and recipe page) and we would either improve or rework the confusing functionality (auto-complete, add ingredients to cart) with the user complaints in mind.

Surprises and What we've learned:

At the start of our project we expected the data we can gather would be clean

Building a site with a lot features planned at the beginning and using a MVC framework are nice to have. However when we have to learn a new framework and a new language, it takes a lot of time to learn how it works and how to build a good secured website without code duplication. We had to cut down on features that are not core to our website. In addition we had to throw scalability and security out the window and moved from our intention to use Django and used PHP, MySQL to render our sites but we still used Python for our backend.

One disappointing technical challenge we ran into was that Amazon Fresh did not have public API so that seamless integration of purchasing ingredients from Amazon Fresh was not possible. Therefore all we can do (at least with the time we had) was to use iframe and find workarounds to make the user able to buy the ingredients on Amazon Fresh.

Future work:

One of the biggest limitations we faced was the limited data we had especially for the images of recipes. Like the saying *"A picture* is worth a thousand *words"*, a good picture of a recipe can get the user interested in the recipe while long description of a recipe makes users to go to a different recipe site. If we could effectively match all the recipes with its appropriate images users can effectively search for the recipes they are searching for so they can decide right away which recipe to cook. We can achieve this by doing more sophisticated table structures and data in the database and also have users themselves update and add images to recipes that are missing pictures as well as submit a recipe to the site.

As we mentioned in the surprised section above, we would like to integrate Amazon Fresh into the website without using somewhat clunky popup frame. In addition to this we can allow users to select only the ingredients that they need from the list of ingredients then a single click of "Buy ingredients" would list the choices for them in Amazon Fresh.

Moreover, we could add a feature for the users to update and add ingredients they already have, then let them search for recipes that contain most of these ingredients. This would enable users to use whatever they already have in their fridges (less food waste) and find a recipe that is new and they might enjoy.

Another feature that we started but didn't complete is using the slope one algorithm for the recommendation of recipes.

Other features started but not completed is to finish the categorization of the recipes on "Quick & Easy", "Seasonal", "Healthy", "International Cooking" and other categories.

Allow the users to rate recipes and display how-to cook videos from the internet especially from YouTube to cook that specific recipe are good features to have.

Also, one of the most important tools to have on our site would be an automatic crawler that periodically goes on all the famous recipes sites and constantly archive and index the newly added recipes, get the images, ratings, videos from YouTube.

We would also translate the recipe site into different languages for users of type of users to use.

Conclusions:

The project's goals were accomplished with the main basic functionality working. The recipes we got from Wikia were over 15,000 and more than 100,000 reviews and over 1,000 images are good enough for any user find their favorite recipes and buy the ingredients on Amazon Fresh a big plus over other recipe sites.

Appendicies

Work distribution

- Pania: Page layout and front-end code (php, js, css, html, sql)
- Nick: Db Schema, translator/exporter glue code
- Min: Screen Scraping, reviews and images.

Group dynamics

Our group had some disagreements about which project to work on initially; afterwards there was some debate over technical choices which didn't cause many problems. We had some motivation problems throughout the quarter but there were no problematic members of the group.

External code

- Mwlib, for parsing xml
- Py-wikimarkup, for rendering mediwiki fragments.
- Various open source jQuery snippets found from tutorials/etc on the web.
- Scrapy, for screen scraping.

Running the project

The steps to run the web application are not particularly streamlined, any instructions we could provide would prove insufficient. You can visit **www.fameme.com/freshipes** to view a live version of our site. If you are interested in running/building the application the basic requirements are:

- Mysql >= 5.1
- Apache >= 2.2

To run the python code you need python >= 2.6 and various eggs installed (mwlib, scrapy, py-wikimarkup). This can be best accomplished using easy_install. Configure the site install by unpacking the site code tarball into the document root. Update the connection string details and then import the data into the database.